(19) KOREAN INTELLECTUAL PROPERTY OFFICE

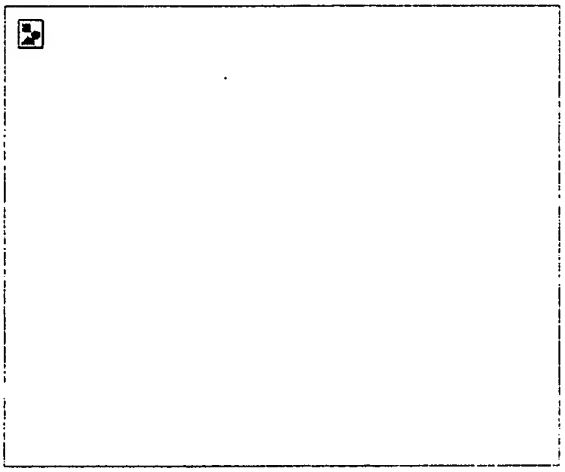
KOREAN PATENT ABSTRACTS

(11)Publication number: 000047570 A
(43)Date of publication of application: 25.07.2000

(21)Application number: 990047499
(22)Date of filing: 29.10.1999
(30)Priority: 30.10.1998 JP 98 311182
02.12.1998 JP 98 343433
(51)Int. Cl G11B 19/02

(71)Applicant: SONY CORPORATION
(72)Inventor: IKARASHI TATSUYA

(54) HIERARCHICAL MANAGEMENT FILE DEVICE, METHOD FOR WRITING AND READING FILE MANAGED HIERARCHICALLY AND ELECTRONIC DEVICE TO EMBODY HIERARCHICAL MANAGEMENT FILE DEVICE



(57) Abstract:

PURPOSE: A hierarchical management file device and an electronic device using it are provided to effectively write various sizes of files and to write and to read large data such as an audio and a video data in a real time by minimizing a loss.

CONSTITUTION: A recording area of an information recording medium is hierarchically divided to areas having recording unit lengths differed each other. Each area of a hierarchy is formed by areas of lower hierarchy under the hierarchy. A file is recorded and played by a software controller referred to a first and a second management information. Areas of larger recording unit lengths are allocated to files played continuously such as a file having an image and sound information. Thereby, an accessing number about the first management information is reduced by reducing a number of areas used for storing the file. Areas having smaller recording unit lengths are allocated to a file played discontinuously such as a sentence. Thereby, the recording area is used efficiently.

Legal Status
Final disposal of an application (application)

(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(51) Int. Cl.
G11B 19/02

(11) 공개번호 특2000-0047570
(43) 공개일자 2000년07월25일

(21) 출원번호 10-1999-0047499
(22) 출원일자 1999년10월29일
(30) 우선권주장 98-311182 1998년10월30일 일본(JP)
98-343433 1998년12월02일 일본(JP)
(71) 출원인 소니 가부시끼 가이샤 이데이 노부유끼
일본국 도쿄도 시나가와쿠 키타시나가와 6초메 7반 35고
(72) 발명자 이가라시타쯔야
일본도꾜도시나가와꾸기다시나가와6쪼메7-35소니가부시끼가이샤내
(74) 대리인 이병호

심사청구 : 없음

(54) 계층적 관리 파일 장치, 계층적으로 관리된 파일의 기록 및 재생 방법, 계층적 관리 파일 장치를 구현하는 전자 장치

요약

본 발명에 따른 계층적 관리 파일 장치와 이를 이용한 전자 장치는 연속적이고 속도가 빠른 기록과 재생이 필요한 대형 파일을 효율적으로 기록하고 재생한다. 대형 파일과 소형 파일은 혼합적으로 매체에 기록된다. 기록 매체의 기록 영역은 클러스터들로 분할되고, 각 클러스터들은 프레그먼트들로 세분된다. 파일은 정보 기록 매체로 계층적으로 기록되고 정보 기록 매체로부터 계층적으로 재생된다. 이 때, 클러스터의 체인과 각 클러스터의 프레그먼트들의 사용 상태를 나타내는 클러스터 FAT와, 프레그먼트들의 연결 상태를 나타내는 프레그먼트 FAT 엔트리와, 클러스터FAT와 프레그먼트 FAT를 정리한 디렉토리 엔트리 테이블을 참조한다.

대표도

도2

색인어

계층적 관리 파일 장치, 프레그먼트, 클러스터, 엔트리

명세서

도면의 간단한 설명

도 1은 계층적 관리 파일 장치와 이 파일 장치를 이용하는 전자 장치를 나타낸 개략도이다.

도 2는 본 발명의 실시예에 따라 정보 기록 매체의 기록 영역 구조를 나타낸 도면이다.

도 3은 본 발명에 따라 클러스터 FAT 엔트리 형태를 보여주는 도표이다.

도 4는 본 발명에 따라 프레그먼트 FAT 엔트리 형태를 보여주는 도표이다.

도 5는 도 3 및 도 4의 클러스터 FAT 엔트리 형태와 프레그먼트 FAT 엔트리 형태를 이용하는 본 발명의 계층적 관리 파일 장치의 동작을 보여주는 도면이다.

도 6은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 파일을 읽는 과정을 보여주는 흐름도의 일부이다.

도 7은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 파일을 읽는 과정을 보여주는 흐름도의 다른 부분이다.

도 8은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 클러스터 단위로 파일을 읽는 과정을 보여주는 흐름도의 일부이다.

도 9는 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 클러스터 단위로 파일을 읽는 과정을 보여주는 흐름도의 다른 부분이다.

도 10은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 프레그먼트 단위로 파일을 읽는 과정을 보여주는 흐름도의 일부이다.

도 11은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 프레그먼트 단위로 파일을 읽는 과정을 보여

주는 흐름도의 다른 부분이다.

도 12는 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 파일을 기록하는 과정을 보여주는 흐름도의 일부이다.

도 13은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 파일을 기록하는 과정을 보여주는 흐름도의 다른 부분이다.

도 14는 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 클러스터 단위로 파일을 기록하는 과정을 보여주는 흐름도의 일부이다.

도 15는 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 클러스터 단위로 파일을 기록하는 과정을 보여주는 흐름도의 다른 부분이다.

도 16은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 새 클러스터를 획득하는 과정을 보여주는 흐름도이다.

도 17은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 검색 영역으로부터 새 클러스터를 획득하는 과정을 보여주는 흐름도의 일부이다.

도 18은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 검색 영역으로부터 새 클러스터를 획득하는 과정을 보여주는 흐름도의 다른 부분이다.

도 19는 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 프레그먼트 단위로 파일을 기록하는 과정을 보여주는 흐름도의 일부이다.

도 20은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 프레그먼트 단위로 파일을 기록하는 과정을 보여주는 흐름도의 다른 부분이다.

도 21은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 새 프레그먼트를 획득하는 과정을 보여주는 흐름도이다.

도 22는 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 검색 영역으로부터 새 프레그먼트를 획득하는 과정을 보여주는 흐름도의 일부이다.

도 23은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에 검색 영역으로부터 새 프레그먼트를 획득하는 과정을 보여주는 흐름도의 다른 부분이다.

도 24는 본 발명에 따라 소프트웨어 관리 수단의 제어 하에, 사용 중인 클러스터로부터 새 프레그먼트를 획득하는 과정을 보여주는 흐름도의 일부이다.

도 25는 본 발명에 따라 소프트웨어 관리 수단의 제어 하에, 사용 중인 클러스터로부터 새 프레그먼트를 획득하는 과정을 보여주는 흐름도의 다른 부분이다.

도 26은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에, 사용 중인 클러스터로부터 새 프레그먼트를 획득하는 과정을 보여주는 흐름도의 또 다른 부분이다.

도 27은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에, 계층 파일의 새 프레그먼트를 획득하는 과정을 보여주는 흐름도의 일부이다.

도 28은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에, 계층 파일의 새 프레그먼트를 획득하는 과정을 보여주는 흐름도의 다른 부분이다.

도 29는 본 발명에 따라 소프트웨어 관리 수단의 제어 하에, 새 파일을 만드는 과정을 보여주는 흐름도의 일부이다.

도 30은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에, 새 파일을 만드는 과정을 보여주는 흐름도의 다른 부분이다.

도 31은 본 발명에 따라 소프트웨어 관리 수단의 제어 하에, 새 파일의 자식 파일을 만드는 과정을 보여주는 흐름도의 일부이다.

도 32는 본 발명에 따라 소프트웨어 관리 수단의 제어 하에, 새 파일의 자식 파일을 만드는 과정을 보여주는 흐름도의 다른 부분이다.

도 33은 퍼스널 컴퓨터의 자기 디스크에 대한 종래의 기록 패턴을 설명하는 도면이다.

도 34는 도 33에 도시한 자기 디스크의 기록 패턴에서 클러스터의 배열을 설명하는 도면이다.

도 35는 도 33에 도시한 자기 디스크의 디렉토리 엔트리의 배열을 설명하는 도면이다.

도 36은 도 33에 도시한 자기 디스크의 FAT 엔트리의 배열을 설명하는 도면이다.

도 37은 종래의 파일 장치에 의한 손실 발생 과정을 설명하는 도면이다.

*도면의 주요 부호에 대한 간단한 설명

501 : CPU　　502 : 메모리

503 : 이차 기록 장치　　504 : 네트워크 드라이버

510 : 계층적 관리 파일 장치

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 계층적 관리 파일 장치와 이를 구현하는 전자 장치, 그리고 계층적으로 관리된 파일의 기록 및 재생 방법에 관한 것이다. 특히, 본 발명은 정보 기록 매체가 복수개의 기록 단위 길이를 갖는 영역으로 나누어지고, 각 계층 수준의 영역이 낮은 계층의 복수 영역으로 이루어지는 계층적 관리 파일 장치에 관한 것이다. 여기서, 파일은 제 1 관리 정보와 제 2 관리 정보에 의하여 계층의 영역에 따라 기록 매체의 파일 기록 영역에 분할 방식으로 기록되고, 제 1 관리 정보와 제 2 관리 정보에 따라 계층의 영역을 계층적으로 관리함으로써 기록 매체로부터 재생된다. 또한, 본 발명은 계층적으로 관리된 파일을 기록하고 재생하는 방법 이외에도, 계층적 관리 파일 장치를 구현하는 전자 장치에 관한 것이다.

하드 디스크, 광디스크 등과 같은 2차 기록 장치를 구비하는 정보 처리 장치에는 정보를 파일에 기초한 기록 영역으로 기록하고 이로부터 재생하는 방식으로 기록 장치의 기록 영역을 관리하는 장치를 장착한다. 이와 같은 장치를 일반적으로 파일 장치라고 한다. 각종 파일 장치들이 개발되어 대형 컴퓨터에 대하여 오랜 기간 동안 사용되어 왔다. 소형 컴퓨터는 퍼스널 컴퓨터라고 하며, 1979년 이후에 널리 보급되었다. 퍼스널 컴퓨터도 플로피 디스크와 하드 디스크를 관리하는 파일 장치를 구비한다. 보통, 이와 같은 퍼스널 컴퓨터의 파일 장치는 FAT(파일 할당 테이블)라고 하는 파일 관리 테이블로 구현된다. FAT는 컴퓨터 장치의 파일 장치뿐만 아니라 각종 전자 제품과 산업 기기의 파일 장치로도 사용된다. FAT는 구성이 간단하므로 퍼스널 컴퓨터나 전자 제품 및 산업 기기의 운영 시스템에 일체화시켜 사용된다.

도 33은 퍼스널 컴퓨터에 사용되는 자기 디스크를 보여준다. 부트 섹터는 운영 시스템을 시동하기 위한 수행 프로그램과, 나중에 설명하게 될 다수의 루트 디렉토리와, 다수의 FAT 등을 저장하기 위한 영역이다. 루트 디렉토리는 기록 파일의 이름, 파일 크기 등을 저장하는 영역이다. 루트 디렉토리와 TAT를 이용하여 원하는 파일로부터 정보를 읽고 원하는 파일로 정보를 기록할 수 있다.

다음은, 도 34, 35 및 36을 참조하여 FAT로 구현되는 파일 장치를 간단히 설명하기로 한다.

먼저 도 34를 보면, '볼륨'이라고 하는 정보 기록 매체의 기록 영역은 주요 기록 단위 길이를 갖는 영역들로 나누어진다. 이들 영역을 '클러스터'라고 한다. 클러스터에는 각각의 클러스터 번호가 부여된다. 정보는 클러스터 단위로 나누어진다. 즉, 정보의 기록과 재생이 클러스터 단위로 이루어진다.

나중에 언급하게 될 클러스터 번호와 도면의 참조 번호를 혼동하지 않도록 하기 위하여, 참조 번호의 사용법에 대하여 설명하면 다음과 같다. 가령, '영역(300)'은 클러스터가 기록되는 영역이나, 복수개의 클러스터가 기록되는 기록 영역, 또는 데이터가 기록되는 클러스터 내의 영역을 나타낸다. 이와 같은 영역은 도면에 나타나 있으며 참조 번호 300번으로 지정된다. 그러므로, 번호 300은 단순히 도면상의 영역을 나타내기 위한 것이다. 반면에, 'FAT 엔트리(300)'은 번호 300이 FAT를 구성하는 단위인 엔트리 중의 하나를 나타내는 것이다.

일반적으로, 클러스터는 512 바이트 내지 32 킬로바이트의 크기를 갖는다. 다음 설명에서 클러스터의 크기는 512 바이트인 것으로 가정한다. 예를 들면, 영역(102)의 클러스터 번호가 20이고 영역(103)의 클러스터 번호가 30이다. 10로 지정된 영역은 전체 정보를 저장하는 시스템 영역이다. 시스템 영역(101)으로부터 클러스터의 총 개수와 클러스터의 크기 등을 알아낼 수 있다.

파일 장치는 '디렉토리'라고 하는 관리 영역을 구비한다. 관리 영역은 복수개의 파일을 포괄적으로 관리한다. 디렉토리는 디렉토리의 관리하에 파일 이름, 파일의 작성 날짜, 파일 크기 등과 같이 파일과 관련한 각종 정보를 저장한다.

도 35는 디렉토리의 구조를 보여 준다. 디렉토리는 파일과 마찬가지고 클러스터 단위로 기록된다. 디렉토리는 루트 디렉토리와 루트 디렉토리를 통해 언급되는 서브 디렉토리를 갖는 계층적인 구조를 이룬다. 디렉토리를 계층적으로 따라 들어가면 소정 파일의 디렉토리 엔트리를 만날 수 있다.

도 35에서 영역(204)에 나와 있는 'FILE2.DAT'는 파일 이름이다. 영역(206)의 파일 크기로부터 알 수 있듯이, 이 파일은 2013 바이트의 크기를 갖는다. 영역(205)은 파일을 구성하는 복수개의 클러스터 중에서 첫 번째 클러스터를 나타내는 클러스터 번호 '4'를 저장한다. 이 클러스터에서 시작되는 FAT 체인을 따라 들어가면, 파일을 구성하는 모든 클러스터에 정보를 읽고 기록할 수 있다.

도 36은 FAT의 구조를 보여 준다. FAT의 각 엔트리는 도 34에 나타난 기록 영역의 각 클러스터에 해당한다. 이들 엔트리는 도 36의 클러스터 2 번 내지 12 번의 순서로 배열된다. 각 FAT의 엔트리는 파일의 다음 구성 클러스터인 클러스터의 클러스터 번호를 보여준다. 가령, 도 35의 디렉토리 엔트리가 나타내는 파일 FILE2.DAT의 첫 번째 클러스터 4(각 클러스터 번호를 갖는 다른 클러스터들과 마찬가지로 클러스터 번호 '4'를 갖는 클러스터)의 FAT 엔트리(305)는 '7'의 값을 갖는다. 이는, 이 파일을 구성하는 클러스터 중에서 다음 클러스터가 클러스터 7이라는 것을 의미한다. 클러스터 7의 FAT 엔트리(308)의 값은 11이고, 클러스터 11의 FAT 엔트리(312)의 값은 9이고, 클러스터 9의 FAT 엔트리(310)의 값은 65535이다. 여기서, 65535는 특별한 의미를 갖는다. 즉, 파일을 구성하는 클러스터들 중에서 가장 마지막 클러스터라는 것이다.

이와 같은 FAT 엔트리 체인으로부터, 파일 FILE2.DAT가 네 개의 클러스터로 구성되는 것을 알 수 있다. 즉, 도 34에 나타낸 디스크 기록 영역(104)의 클러스터 4, 영역(107)의 클러스터 7, 영역(111)의 클러스터 11, 그리고 영역(109)의 클러스터 9가 파일을 구성한다. 한편, 영역(206)은 파일 크기가 2013이라는 것을 나타낸다. 각 클러스터의 크기가 512 바이트이므로, 마지막 클러스터 9의 477 바이트만 사용되는 것을 알 수 있다(2013 - 512 x 3 = 477). 영역(306), 영역(311) 및 영역(313)의 FAT 엔트리 값들은 '0'이다. 이는, 이들 영역의 클러스터가 사용되지 않고 있다는 것을 의미한다. 그러므로, FAT 엔트리 값의 영

역(306)(311)(313)에 대응하는 영영(105)(110)(112)의 클러스터 5, 클러스터 10 및 클러스터 12는 새로 기록하거나 새로운 파일을 만들었을 때 파일의 크기가 증가하는 경우에 새로운 기록 영역으로 사용된다.

FAT에 의한 관리는 간단하지만 다음과 같은 문제점이 있다.

(1) 파일에 연속 클러스터가 사용되는 경우에도 클러스터의 FAT 엔트리를 하나씩 참조해야 한다. 따라서, FAT 엔트리에 접근하는 횟수가 많아서 파일에 임의 접근시 효율성이 떨어진다.

(2) 파일에 고속 접근하기 위해서는 파일의 데이터를 가능한 연속 클러스터에 지정해야 한다. 엔트리 클러스터도 각각의 FAT 엔트리에 대응하므로, 연속적인 빈 클러스터의 배치를 찾아내기 위해서는 빈 클러스터를 하나씩 참조해야 한다. 따라서, FAT 엔트리에 접근하는 횟수가 증가한다. 또한, 원치 않는 손실을 최소화하도록 파일을 연속 클러스터에 효율적으로 지정하는 것이 불가능하다. '손실'이라 함은 기록 영역이 지나치게 많은 범위로 나누어진 것을 말한다. 여기서, '범위'라 함은 복수개의 연속 클러스터로 구성되는 단위를 말한다.

(3) 기록 장치의 규모가 증가하는 최근의 추세에 따라서 FAT의 크기가 증가하였다. 이는, FAT 엔트리를 이용하기 위해서 보다 많은 메모리 용량과 보다 높은 CPU의 처리 속도를 요구한다. 결과적으로, 하드웨어의 이용 효율이 제한적이기 때문에 손실을 내지 않기 위한 파일 할당이 이루어지지 않게 된다.

(4) 최근의 기술 발전으로 컴퓨터를 이용한 각종 데이터의 다양화가 이루어졌다. 요즘에는, 워드프로세서로 작성한 텍스트와 같은 수 킬로바이트의 소형 파일들과 오디오 및 비디오 데이터를 포함하는 파일과 같은 수 기가 바이트의 대형 파일들이 하나의 기록 매체에 저장된다. 이와 같이 하나의 기록 매체에 여러 크기의 파일들이 당겨지면 손실이 발생하기 쉽다. 보다 작은 크기의 클러스터를 이용함으로써 기록 영역을 이용 효율을 높일 수 있다. 그러나, 이와 같은 경우, 파일이 너무나 많은 영역으로 나누어지므로 데이터의 전송률이 떨어져서 오디오 및 비디오 데이터의 실시간 기록 및 재생이 어려워진다.

도 37은 데이터 전송률이 감소하는 예를 나타낸 것이다. 영역(401)(403)(405)(407)의 클러스터들은 불연속적으로 배열되어 있다. 그러므로 각 클러스터의 데이터를 읽어들이는 데에 필요한 시간 이외에 영역(402)(404)(406)이 지정하는 탐색 시간이 필요하다. 이와 같이 탐색 시간이 필요해짐에 따라 데이터 전송률이 감소하게 된다.

가능한 한 많은 클러스터를 관리하기 위하여 32비트로 크기가 증가된 FAT 엔트리를 이용하는 확장 FAT를 구현하는 방법이 제안되었다. 그러나, 이와 같은 방법은 상기에 언급한 문제들을 해결하지 못한다. 또한, 크기가 작은 클러스터를 이용함으로써 기록 매체의 기록 영역을 효율적으로 이용할 수 있도록 하는 방법이 제안되었다. 그러나, 이 방법은 오디오/비디오 파일과 같은 대형 파일을 기록할 때에 특히 손실을 발생하는 단점이 있다. 이 문제에 대한 방안으로서, '손실제거프로그램'으로 알려진 소프트웨어를 사용하여 파일을 기록한 후에 클러스터 데이터를 다시 재할당하고 기록하는 방법에 제안되었다.

앞서 설명한 바와 같이, 본 발명의 목적은 간단한 FAT 관리 기술을 효율적으로 이용하고, 큰 할당 단위를 갖는 클러스터 FAT와 작은 할당 단위를 갖는 프레그먼트 FAT를 이용하여 기록 영역을 계층적으로 이용하여 상기 (1) 내지 (4)의 문제점을 극복하여 각종 크기의 파일들을 효율적으로 기록할 수 있으면서도 손실을 최소화하여 오디오 및 비디오 데이터와 같은 크기가 큰 데이터의 실시간 기록 및 재생을 가능케 하는 계층적 관리 파일 장치와 이를 이용한 전자 장치를 제공하는 것이다.

발명이 이루고자하는 기술적 과제

상기 목적을 달성하기 위하여, 본 발명에 일 측면에 따르면, 정보 기록 매체의 기록 영역이 복수개의 기록 단위 길이를 갖는 영역으로 계층적으로 분할되고, 각 계층의 영역이 하위 계층의 복수개의 영역으로 이루어지는 계층적 관리 파일 장치가 제공된다. 여기서, 파일은 제 1 관리 정보와 제 2 관리 정보의 관리하에 계층들의 영역 단위로 기록 매체의 파일 기록 영역에 분할 방식으로 기록된다. 그리고, 파일은 제 1 관리 정보와 제 2 관리 정보에 따라 계층들의 영역을 계층적으로 관리함으로써 기록 매체로부터 재생된다. 상기 계층적 관리 파일 장치는 정보 기록 매체와 소프트웨어 제어 수단을 포함한다. 정보 기록 매체는 제 1 관리 정보와 제 2 관리 정보를 기록하고, 제 1 관리 정보와 제 2 관리 정보의 재생을 가능케 한다. 제 1 관리 정보는 각 계층에 대하여 각 계층의 영역의 연결 상태를 나타내는 연결 정보와 영역의 사용 상태를 나타내는 사용 정보를 포함하고, 상기 제 2 관리 정보는 상기 파일의 속성, 파일의 기록 및 재생에 사용되는 계층의 등급, 파일의 형태, 그리고 파일의 기록 및 재생이 시작되는 시작 계층의 영역 위치를 포함한다. 상기 소프트웨어 제어 수단은 각 계층의 제 1 관리 정보와 제 2 관리 정보를 참조하여 파일의 기록 및 재생을 관리하고, 상기 파일을 각 계층들의 영역에 기록하기 위하여 하위 계층의 제 1 관리 정보를 참조하여 하위 계층의 영역들을 파일에 할당하고; 하위 계층에 빈 영역이 존재하지 않으면 상위 계층의 제 1 관리 정보를 이용한 후에 상위 계층에 속하는 하위 계층의 제 1 관리 정보를 이용하여 상기 영역들에 파일을 나누도록 하고; 파일의 특성에 따라 계층을 선택적으로 이용한다.

바람직하게는, 본 발명의 계층적 관리 파일 장치는 다음 (1) 내지 (13) 중의 어느 한 가지의 형태를 갖는다.

(1) 소프트웨어 제어 수단은 각 계층에 주어진 번호가 증가하는 순서대로 제 1 관리 정보, 제 2 관리 정보 및 상기 파일 기록 영역을 각 계층의 영역들에 기록 및 재생한다. 이렇게 하여, 상기 영역들에는 오름차순의 번호가 부여된다.

(2) 소프트웨어 제어 수단은 제 2 관리 정보를 같은 계층의 영역들에 기록하도록 제어한다. 이 때, 제 2 관리 정보는 적어도 상기 파일의 속성들을 나타내는 제 1 종의 정보와, 파일의 기록 및 재생에 이용되는 계층과 파일의 형태를 나타내는 제 2 종의 정보와, 파일의 기록 및 재생이 시작되는 계층의 영역들의 위치를 나타내는 제 3 종의 정보를 포함한다. 상기 소프트웨어 제어 수단은, 제 2 종의 정보에 따라 상기 파일이 계층 파일이 아니라고 판단되면, 제 3 종의 정보와 결정된 계층을 이용하여 상기 파일의 기록 및 재생을 수행한다.

(3) 소프트웨어 제어 수단은 제 2 관리 정보를 같은 계층의 영역들에 기록하도록 제어한다. 이 때, 제 2 관리 정보는 적어도 상기 파일의 속성들을 나타내는 제 1 종의 정보와, 파일의 기록 및 재생에 이용되는 계층과 파일의 형태를 나타내는 제 2 종의 정보와, 파일의 기록 및 재생이 시작되는 계층의 영역들의 위치를 나타내는 제 3 종의 정보를 포함한다. 제 2 종의 정보에 따라 상기 파일이 부모 파일과 자식 파일로 구성되는 계층 파일이라고 판단되면, 상기 소프트웨어 제어 수단은 부모 파일의 계층의 시작 영역의 위치를 나타내는 부모 파일의 제 3 종의 정보와 일치하는 자식 파일의 제 2 종의 정보를 검색하여 자식 파일을 결정하고, 자식 파일의 제 3 종의 정보에 따라 상기 자식 파일의 기록 및 재생이 시작되는 계층의 영역의 위치를 결정한 다음에, 결정된 자식 파일의 제 1 관리 정보를 이용하여, 자식 파일이 사용한 하위 계층의 영역들이 부모 파일이 사용한 상위 계층의 영역에 반드시 포함되도록 제어한다.

(4) 소프트웨어 제어 수단은, 제 1 관리 정보가 각 계층의 영역들의 연결 상태를 나타내는 연결 정보와 각 계층의 영역들의 사용 상태를 나타내는 사용 정보를 갖는 제 4 종의 정보와, 그리고 상기 계층보다 낮은 계층의 영역들의 사용 상태를 나타내는 제 5 종의 정보를 포함하도록 한다. 이 때, 제 4 종의 정보에 따라 상기 계층의 영역 다음에 기록 및 재생에 사용될 영역을 결정하여 상기 파일에 할당된 영역들을 기록 및 재생에 연속적으로 사용할 수 있도록 제어한다. 제 5 종의 정보에 따라 하위 계층의 영역을 파일에 할당할지를 결정한다.

(5) 하위 계층의 제 1 관리 정보는 하위 계층의 영역들의 연결 상태를 나타내는 연결 정보와, 하위 계층의 영역들의 사용 상태를 나타내는 사용 정보를 포함하고; 또한, 하위 계층의 상기 영역들이 속하는 상위 계층의 영역들의 연결 상태를 나타내는 연결 정보와, 상위 계층의 영역들의 사용 상태를 나타내는 사용 정보를 포함한다.

(6) 각 계층의 제 1 관리 정보는 상기 파일의 종료점을 나타내는 정보를 포함한다.

(7) 소프트웨어 제어 수단은, 정보 기록 매체가 초기화되면 가장 큰 기록 단위 길이를 갖는 최상위 계층의 영역들만을 할당하도록 제어한다.

(8) 소프트웨어 제어 수단은 상기 파일의 연속적인 정보 단위를 하위 계층의 영역들에 부여하도록 동작한다.

(9) 임의 계층의 새로운 영역이 임의 파일에 대하여 보존되는 경우와 상기 파일이 새로운 영역에 기록되는 경우, 소프트웨어 제어 수단은 상기 계층의 영역들에 대한 연결 정보 및 사용 정보를 갖는 제 4 종의 정보와 하위 계층의 영역들의 사용 상태를 나타내는 제 5 종의 정보를 각 계층에 대하여 갱신한다.

(10) 임의 계층의 새로운 영역이 임의 파일에 대하여 보존되는 경우와 상기 파일이 새로운 영역에 기록되는 경우, 소프트웨어 제어 수단은 상기 파일의 속성을 나타내는 제 1 종의 정보를 갱신하고 제 2 관리 정보에 포함된 제 2 종의 정보와 제 3 종의 정보를 갱신한다. 이 때, 제 1 종의 정보는 상기 파일의 기록 및 재생에 사용되는 계층을 나타내고, 제 2 종의 정보는 파일의 기록 및 재생이 시작되는 계층의 영역의 위치를 나타낸다.

(11) 소프트웨어 제어 수단은 파일을 기록하기 위한 새로운 영역을 할당하여 제 1 관리 정보와 제 2 관리 정보를 갱신할 때에는 제 1 관리 정보와 제 2 관리 정보를 참조하고, 파일이 가장 최근에 기록된 현재 계층의 영역의 위치를 이용하여 제 1 관리 정보와 제 2 관리 정보를 갱신할 때에는 현재 계층의 영역이 속하는 최상위 계층의 영역의 제 1 관리 정보를 참조하여 전체 계층 중에서 최상위 계층의 사용하지 않은 영역을 검색한다. 소프트웨어 수단은 연속적인 하위 계층에 대하여 최상위 계층의 영역의 검색과 유사한 검색을 수행하고, 파일에 첨가될 정보를 현재의 계층의 영역에 가장 가까운 검색 영역으로 할당한다.

(12) 소프트웨어 제어 수단은 계층 파일을 기록하기 위한 새로운 영역을 할당하여 제 1 관리 정보와 제 2 관리 정보를 갱신할 때에는 제 1 관리 정보와 제 2 관리 정보를 참조하고, 계층 파일이 가장 최근에 기록된 현재 계층의 영역의 위치를 이용하여 제 1 관리 정보와 제 2 관리 정보를 갱신할 때에는 현재 계층의 영역이 속하고 부모 파일에 속하는 최상위 계층의 영역의 제 1 관리 정보를 참조하여 전체 계층 중에서 부모 파일에 속하는 최상위 계층의 사용하지 않은 영역을 검색한다. 소프트웨어 수단은 연속적인 하위 계층에 대하여 부모 파일에 속하는 최상위 계층의 사용하지 않은 영역의 검색과 유사한 검색을 수행하고, 현재 계층의 영역에 가장 가까운 검색 영역에 상기 계층의 자식 파일의 정보를 기록한다.

(13) 각 계층에 대하여 제 1 관리 정보는 각 계층의 영역들의 사용 상태를 나타내는 사용 정보와, 상기 계층보다 낮은 계층의 영역들의 사용 상태를 나타내는 사용 정보를 포함한다. 각 계층의 영역들의 연결 상태를 나타내는 연결 정보는 제 2 관리 정보나 제 1 관리 정보와 제 2 관리 정보가 기록되어 있는 영역과 다른 기록 영역에 주어진 제 3 관리 정보에 포함된다.

본 발명의 또한 상기 계층적 관리 파일 장치를 이용한 전자 장치도 제공한다.

본 발명의 또 다른 측면에 따르면, 계층적으로 관리된 파일을 정보 기록 매체의 기록 영역에 기록하고 이로부터 재생하는 방법을 제공한다. 여기에서, 상기 기록 영역은 복수개의 기록 단위 길이를 갖는 영역들로 계층적으로 분할되고, 각 계층의 영역은 하위 계층의 복수개의 영역들로 이루어진다. 파일은 제 1 관리 정보와 제 2 관리 정보의 관리하에 계층의 영역 단위로 기록 매체의 파일 기록 영역에 분할 방식으로 기록되고, 제 1 관리 정보와 제 2 관리 정보에 따라 계층의 영역들을 계층적으로 관리함으로써 기록 매체로부터 재생된다. 상기 정보 기록 매체는 제 1 관리 정보와 제 2 관리 정보를 기록한다. 각 계층에 대하여, 제 1 관리 정보는 각 계층의 영역들의 연결 상태를 나타내는 연결 정보와 영역들의 사용 상태를 나타내는 사용 정보를 포함하고, 제 2 관리 정보는 상기 파일의 속성들과, 파일의 기록 및 재생에 사용되는 계층의 등급과, 파일의 기록 및 재생이 시작되는 시작 계층의 영역의 위치를 포함한다. 상기 방법은 각 계층의 제 1 관리 정보와 제 2 관리 정보를 이용하여, 상기 파일을 계층의 영역들에 기록하기 위하여 하위 계층의 제 1 관리 정보를 참조하여 상기 파일의 하위 계층의 영역들을 할당하는 단계와; 그리고 하위 계층의 빈 영역이 존재하지 않으면, 상위 계층의 제 1 관리 정보를 참조한 후에, 상위 계층에 속하는 하위 계층의 제 1 관리 정보를 탐하여 다시 검색을 수행하여 상기 영역들에 파일을 할당하고; 파일의 특성에 따라 계층을 선택적으로 이용하면서 정보 기록 매체로 파일을 기록하고 정보 기록 매체로부터 파일을

재생하는 단계를 포함한다.

상기와 같은 특징을 갖는 본 발명은 다음의 이점을 갖는다.

(1) 정보 기록 매체의 기록 영역이 서로 다른 기록 단위 길이를 갖는 영역들로 계층적으로 분할된다. 계층의 각 영역은 상기 계층 아래에 존재하는 하위 계층의 영역들로 구성된다. 파일은 제 1 관리 정보와 제 2 관리 정보를 참조하는 소프트웨어 제어 수단에 의해 기록되고 재생된다. 가령, 보다 큰 기록 단위 길이의 영역들은 영상 및 음향 정보를 담고 있는 파일과 같은 연속 재생이 필요한 파일들에 할당된다. 이로써, 파일을 저장하는 데에 사용되는 영역의 개수가 감소하여 제 1 관리 정보에 대한 접근 횟수가 줄어든다. 이와 반대로, 보다 작은 기록 단위 길이를 갖는 영역들은 데이터 크기가 작아서 문장과 같이 불연속적으로 재생되는 파일에 할당되어 정보 기록 매체의 기록 영역을 효율적으로 이용할 수 있다.

(2) 파일에 대한 억세스 속도를 높이기 위해서는, 파일의 조각들을 연속 영역에 기록하는 것이 바람직하다. 본 발명에 따르면, 정보 기록 매체의 기록 영역이 복수개의 기록 단위 길이를 갖는 영역들로 계층적으로 분할된다. 각 계층의 영역은 각 계층의 바로 아래에 존재하는 계층의 영역들로 구성된다. 상위 계층의 영역은 하위 계층의 영역의 사용에 관한 정보를 갖는다. 그러므로, 파일을 영역에 할당할 때, 사용하지 않은 영역을 계층적으로 검색하여 손실을 방지하면서도 파일을 집중적으로 고속으로 기록할 수 있다. 이와 같은 이유로, 기록된 파일들을 고속으로 재생할 수 있다.

(3) 상기에 설명한 바와 같이, 보다 큰 기록 단위 길이를 갖는 영역들이 영상 및 음향 정보를 담고 있는 파일과 같이 연속 재생을 요하는 파일에 할당되고, 보다 작은 기록 단위의 길이를 갖는 영역들이 데이터 크기가 작아서 문장과 같이 불연속적으로 재생할 수 있는 파일에 할당된다. 이렇게 함으로써 제 1 관리 정보의 크기가 크게 감소된다. 이 특징은 상기에서 설명한 (1)과 (2)의 이점과 함께 본 발명에 따른 계층적 관리 파일 장치를 이용한 전자 장치의 하드웨어에 가해지는 부하를 크게 줄일 수 있다.

도면의 구성 및 작용

이제부터, 본 발명에 따른 계층적 관리 파일 장치와 이를 이용하는 전자 장치의 실시예에 대하여 설명하기로 한다. 앞으로 설명하게 될 실시예의 계층적 관리 파일 장치는 두 가지 계층 즉, 상위 계층과 하위 계층을 갖는 계층적 파일을 이용한다. 상위 계층은 앞서 언급한 클러스터로 분할되고, 낮은 계층은 나중에 설명하게 될 '프레그먼트'라고 하는 영역으로 분할된다. 클러스터 FAT는 상위 계층의 제 1 관리 정보로 사용되고, 프레그먼트 FAT는 낮은 계층의 제 1 정보로 사용된다. 클러스터가 구성하는 디렉토리 엔트리 테이블은 제 1 관리 정보로 이용된다. 여기서는, 상기와 같은 형태의 계층적 관리 파일 장치와 이를 이용한 전자 장치에 대해서 설명하고 있지만, 이에 한정된 것이 아니며, 본 발명은 다수의 계층을 갖는 계층 파일을 이용하는 다른 형태의 계층적 관리 파일 장치와 이를 이용한 전자 장치에도 적용될 수 있음은 자명하다.

도 1은 본 발명을 구현하는 계층적 관리 파일 장치의 실시예와 이를 이용한 전자 장치를 도시한 개략도이다.

CPU(501)는 고속 반도체 메모리인 메모리(502)에 저장된 프로그램에 따라 시스템을 제어한다. 프로그램은 ROM(도시되어 있지 않음)에 미리 기록된다. 또는, 프로그램은 이차 기록 장치(503)이나 네트워크 드라이버(504)를 통하여, 또는 튜너(509)로부터 메모리(502)로 전송되어 실행된다. 이차 기록 장치(503)로는 하드 디스크, 교환가능 광디스크 또는 플래시 ROM과 같은 비휘발성 메모리가 사용된다. 이차 기록 장치(503)의 정보 기록 매체에 저장된 파일들은 메모리(502)에 저장된 프로그램에 따라 계층적 관리 파일 장치에 의해 관리된다. CPU(501)와 메모리(502)는 청구항 1에 따른 본 발명의 가장 일반적인 형태로 사용되는 소프트웨어 제어 수단을 제공한다. 본 발명의 계층적 관리 파일 장치(510)는 이차 기록 장치(503)와, 소프트웨어 제어 수단과, 정보 기록 매체를 포함한다.

네트워크 드라이버(504)는 이차 기록 장치(503)에 기록된 파일의 데이터를 전송하고 이차 기록 장치(503)에서 읽어들인 파일 데이터를 전송하기 위하여 참조 번호 505로 지정되는 IEEE 1394나 에더넷과 같은 네트워크를 지원한다. 이들 데이터는 메모리(502)에 저장된 프로그램에 따라 CPU나 버스(506)에 연결된 별도의 하드웨어를 이용하여 압축이나 압축 해제와 같이 처리된 후에 네트워크로 전송되도록 되어 있다.

적외선 송수신기(507)는 사용자가 작동하는 원격 제어기로부터 명령을 받아서 메모리(502)에 저장된 프로그램의 동작을 제어하여 상호 운영을 가능케 한다. 영상/음향 입출력 장치(508)는 텔레비전이나 모니터와 같은 표시 장치에 연결되어 메모리(502)에 저장된 프로그램에 따라 그래픽 사용자 인터페이스(도시되어 있지 않음), 이차 기록 장치(503) 및 네트워크 드라이버(504)로부터 전송된 영상과 음향을 출력시킨다.

영상/음향 입출력 장치(508)는 비디오 카메라나 비디오 데크와 같은 장치에 연결되어 비디오 데크로부터 음향과 영상을 픽업하고 비디오 데크에 영상과 음향을 기록한다. 튜너(509)는 위성 방송파나 지상파를 수신하기 위한 안테나에 연결된다. 수신 데이터는 영상/음향 입출력 장치(508)로 전송되고 영상/음향 입출력 장치(508)에 의해 재생된다.

이들 데이터는 먼저 이차 기록 장치(503)에 저장된다. 영상/음향 입출력 장치(508)와 튜너(509)에 의해 처리되는 입출력 신호는 아날로그 신호나 디지털 신호이다. 아날로그 신호는 영상/음향 입출력 장치(508)와 튜너(509)에 의해 디지털 데이터로 변환된다. 이와 같은 디지털 데이터는 압축, 데이터 변환 등을 거쳐서 도 1에 나와 있는 본 발명에 따른 계층적 관리 파일 장치나, 전자 장치(이후부터, '시스템 장치'로 칭함)에서 처리된다.

CPU(501), 메모리(502), 이차 기록 장치(503), 네트워크 드라이버(504), 적외선 송수신기(507), 영상/음향 입출력 장치(508) 및 튜너(509)는 버스(506)에 연결되어 시스템을 구성한다. 도 1에 도시한 시스템 장치는 오직 본 발명의 설명하기 위한 것으로서, 시스템은 네트워크 드라이버(504), 적외선 송수신기(507), 영상/음향 입출력기(508) 및 튜너(509)를 생략할 수 있다. 시스템은 또한 이차 기록 장치(503)도 생략할 수 있다. 이 경우, 본 발명의 파일 장치는 원거리 시스템에 연결되어 있는 다른 이차 기록 장치(503)에 파일을 기록하도록 동작할 수 있다. 또한, 부가 장치 또는 장치들을 상기 시스템에 연결시킬 수 있다.

도 2는 계층적 관리 파일 장치(510)가 수행하는 볼륨 관리 방법을 설명하는 도면이다. FAT를 이용한 종래의 파일 장치의 경우와 같이, 이차 기록 장치(503)의 기록 영역의 볼륨은 일정한 크기의 클러스터들로 나누어지고 클러스터 단위로 관리된다. 2부터 시작되는 정수로 나타낸 일련의 클러스터 번호들이 클러스터에 부여된다.

본 발명에 따라, 각 클러스터는 프레그먼트라고 부르는 보다 작은 크기의 단위로 나누어진다. 클러스터의 프레그먼트에는 일련 번호가 부여된다. 상기 실시예에서, 클러스터의 크기는 64 킬로바이트이고, 프레그먼트의 크기는 512 바이트이다. 그러므로, 각 클러스터는 128개의 프레그먼트들로 이루어지고, 이들 프레그먼트에는 0에서 127까지의 번호가 부여된다.

본 발명의 계층적 관리 파일 장치(510)는 파일을 클러스터 단위로 관리하는 클러스터 FAT와 파일을 프레그먼트 단위로 관리하는 프레그먼트 FAT를 이용한다. 클러스터 FAT는 볼륨의 모든 클러스터들의 FAT 엔트리를 기록하고, 프레그먼트 FAT는 볼륨 내의 클러스터 개수와 각 클러스터 내의 프레그먼트 개수를 곱한 것과 동일한 개수의 FAT 엔트리를 기록한다.

앞서 설명한 바와 같이, 클러스터 FAT는 클러스터 엔트리를 클러스터 번호에 따라 각각 검출하도록 구성된다. FAT는 이차 기록 장치(503)의 클러스터에 기록되고, FAT의 위치 즉, 클러스터 번호는 시스템 영역에 나타나 있다. 높은 효율을 얻기 위해서는, FAT의 일부를 읽고 메모리(502)에 저장하고, 필요한 경우, 이차 기록 장치(502)에 다시 기록하여 데이터를 갱신한다.

도 3은 클러스터 FAT의 엔트리 형태를 설명하고 있다. 클러스터 FAT의 각 엔트리에서, 해당 클러스터의 이용 방법에 따라 다음의 다섯 가지 형태로 32비트값이 기록된다:

1) 사용하지 않은 클러스터

2) 프레그먼트 단위로 사용중인 클러스터

3) (모든 프레그먼트가 사용되고 있는 경우) 프레그먼트 단위로 사용중인 클러스터

4) 파일에 할당된 클러스터

5) (계층 파일에 프레그먼트 단위로 사용중인) 파일에 할당된 클러스터

사용하지 않은 클러스터의 경우, 식별값 '0'이 영역(701)을 구성하는 상위 25 비트에 저장된다. 이는, 클러스터가 사용되지 않았으므로 2) 내지 5)의 형태로 할당될 수 있음을 의미한다. 전체 볼륨이 초기화되면, 모든 클러스터 FAT의 엔트리들이 사용되지 않은 상태로 초기화된다.

클러스터가 프레그먼트 단위로 사용 중인 경우, 식별값 '0'이 영역(702)을 구성하는 상위 25 비트에 저장되고, 클러스터 내에서 사용 중인 프레그먼트의 번호가 영역(703)을 구성하는 하위 7 비트에 저장된다. 클러스터의 사용 상태에 관한 정보는 프레그먼트 FAT를 이용하지 않고도 사용중인 프레그먼트의 번호로부터 알아 낼 수 있으므로, 이 정보에 따라 프레그먼트를 효율적으로 파일에 할당할 수 있다. 예를 들면, 많은 개수의 프레그먼트를 할당할 때, 사용중인 프레그먼트를 갖는 선택함으로써 특정 영역에 파일의 프레그먼트를 집중시킬 수 있으므로 고속 억세스가 가능하다.

3)의 형태는 2)의 형태에서 클러스터의 모든 프레그먼트들이 사용중인 특수한 경우이다. 이 때, 식별값 '1'이 영역(704)을 구성하는 상위 25 비트에 저장되고, 식별값 '0'이 하위 7 비트에 저장된다.

2)의 형태와 3)의 형태 모두에서, 기록 영역은 프레그먼트 단위로 할당되고, 이 파일을 구성하는 모든 프레그먼트들이 프레그먼트 FAT의 FAT 엔트리에 의해 연결된다.

4)의 형태에서와 같이 클러스터가 파일에 할당되는 경우, 파일을 구성하는 클러스터의 다음 클러스터의 번호는 영역(706)을 이루는 상위 25 비트에 기록되고, 식별값 '0'이 영역(707)을 이루는 하위 7 비트에 기록된다. 다음 클러스터가 없으면, 즉 현재의 클러스터가 마지막 클러스터이면, 식별값 33,554,431이 영역(706)을 이루는 상위 25 비트에 저장된다.

4)의 형태와 같이 클러스터가 파일에 할당되었으면, 볼륨의 기록 영역은 클러스터 단위로 파일에 할당된다. 파일을 구성하는 모든 클러스터들이 클러스터 FAT의 FAT 엔트리들에 의해 연결된다.

5)의 형태는 본 발명에 따른 계층적 관리 파일 장치의 장점을 가장 잘 나타낸다. 볼륨의 기록 영역은 클러스터 단위로 파일에 할당된다. 할당된 클러스터는 다른 파일 또는 파일들에 할당 가능한 프레그먼트들로 분할된다. 이와 같은 형태의 파일 구조를 계층 파일이라고 한다. 계층 파일에서, 클러스터 단위로 관리되는 파일을 '부모 파일'이라고 하고, 프레그먼트 단위로 관리되는 파일을 '자식 파일'이라고 한다. 이와 같은 계층 파일을 이용함으로써 자식 파일들을 한 곳에 집중적으로 효율적으로 기록할 수 있으므로, 다수의 자식 파일들을 연속으로 읽어들이거나 데이터를 자식 파일들에 연속으로 기록할 때에 접근시간이 크게 단축된다.

프레그먼트의 할당은 연속적인 네 개의 프레그먼트들을 하나의 단위로 사용하도록 하고, 클러스터를 이들 프레그먼트들을 파일에 단위별로 할당하는데 사용하도록 한다.

상기 4)의 형태에서, 부모 파일의 다른 클러스터의 번호는 도 3에 도시된 영역(708)을 이루는 상위 25 비트에 기록되고, 클러스터에 사용중인 프레그먼트의 개수는 영역(709)을 이루는 하위 7 비트에 기록된다. 모든 프레그먼트들이 사용중일 때에는, '0'이 하위 7 비트에 기록된다. 하위 7 비트는 4)의 형태와 5)의 형태 모두에서 '0'으로 설정된다. 그러나, 이 파일 장치에서는 디렉토리에 따라 파일이 계층 파일인지 아닌지를 결정하여야 관리가 가능하다. 이는, 프레그먼트 FAT의 엔트리들을 검사하여 기록 영역이 클러스터로 사용되고 있는지를 판단하면 된다.

도 4는 프레그먼트 FAT 엔트리의 형태를 보여 준다. 프레그먼트 FAT의 엔트리에서, 32비트값이 프레그먼트의 사용 상태에 따라 다음의 세 가지 형태로 기록된다.

1) 사용하지 않은 프레그먼트

2) 클러스터 단위로 사용중인 프레그먼트

3) 파일에 할당된 프레그먼트

1)의 형태에서, 프레그먼트를 포함하고 있는 클러스터가 프레그먼트 단위로 사용된다. 그러므로, 1)의 형태는 도 3의 클러스터 FAT 엔트리 형태 2), 3) 및 4)에 대응되고, 프레그먼트는 파일에 할당되지 않았다. 이 경우, 식별값 '0'이 영역(801)에 기록된다.

2)의 형태는 프레그먼트를 포함하고 있는 클러스터가 프레그먼트 단위가 아닌 클러스터 단위로 사용되고 있는 것을 나타낸다. 이는 도 3의 클러스터 FAT 엔트리 형태 1)과 4)에 대응되는 것으로, 이 클러스터가 클러스터 FAT에 의해 관리되고 있음을 나타낸다.

식별값 '1'이 영역(802)을 이루는 상위 25 비트에 저장되고, 식별값 '0'이 영역(803)을 이루는 하위 7 비트에 저장된다. 이 클러스터에 속하는 모든 프레그먼트들의 엔트리들은 도 4에 도시한 '2. FRAGMENT IS USE ON CLUSTER BASIS'의 형태를 갖는다. 볼륨을 초기화하면, 모든 클러스터들이 클러스터 단위로 사용되고, 모든 프레그먼트 FAT의 엔트리들이 초기화되어 프레그먼트들이 클러스터 단위로 관리됨을 나타낸다. 이와 같은 초기화 특징은 청구항 8에 따른 본 발명의 한 형태이다.

3)의 형태는, 1)의 형태에서와 같이 클러스터가 프레그먼트 단위로 사용중이고 프레그먼트가 파일에 할당된 경우이다. 파일을 구성하는 다음 프레그먼트는 프레그먼트 FAT의 엔트리에 기록된다. 그러므로, 프레그먼트가 속하는 클러스터의 번호가 영역(804)을 이루는 상위 25 비트에 저장되고, 클러스터 내의 다른 프레그먼트와 구별되는 프레그먼트 번호가 하위 7 비트에 저장된다.

다음은, 도 5A와 5B를 참조하여 상기에서 설명한 클러스터 FAT와 프레그먼트 FAT의 엔트리 형태에 따른 본 발명의 계층적 관리 파일 장치(510)의 동작 상태를 설명하기로 한다. FAT 엔트리는 크기가 32 비트이다. 편의상, 상위 25 비트와 하위 7 비트를 몸마 ','로 구별하고, 숫자에는 부가 표시 'h'를 부여하여 6진법으로 표시한다.

도 5A의 디렉토리는 FILE1.DAT를 저장하는 영역(901)과, FILE2.DAT를 저장하는 영역(905)과, FILE5.DAT를 저장하는 영역(917)과, FILE6.DAT를 저장하는 영역(921)과, FILE7.DAT를 저장하는 영역(925)과, FILE8.DAT를 저장하는 영역(929)을 갖는다.

이 디렉토리는 파일의 경우와 같이 볼륨 내의 클러스터나 프레그먼트에 저장된다. 또한, 디렉토리는 낮은 계층의 디렉토리 즉, 계층 디렉토리의 서브 디렉토리를 저장하므로, 시스템 영역이 나타내는 루트 디렉토리를 통하여 계층 디렉토리를 따라 가면 소정 파일의 디렉토리 엔트리를 검색할 수 있다.

디렉토리 엔트리는 각 파일의 속성을 나타내는 필드를 갖는다. 이들 필드는 파일 이름(영역 901, 905, 909, 913, 917, 925 및 929); 할당 형태(영역 902, 906, 910, 914, 918, 922, 926 및 930); 제 1 계층의 구분 영역인 파일의 선두 종료 위치(영역 903, 907, 911, 915, 919, 923, 927 및 931); 그리고 파일 크기(영역 904, 908, 912, 918, 920, 924, 928 및 932)이다. 파일 이름의 필드에는 256개의 문자에 해당하는 256 바이트가 부여된다. 파일 이름과 파일 크기는 청구항 10에 따라 파일 속성을 구현한다. 파일의 선두 종료 위치는 청구항 10에 기재한 바와 같이 제 1 계층의 영역의 위치이다. 앞서 설명한 바와 같이, 도 5A에 도시한 디렉토리 엔트리 테이블이 청구항 10에 따른 제 2 관리 정보의 실시예이다.

할당 형태의 필드에는 4 바이트가 부여된다. 값 '00000000h'는 볼륨이 클러스터 단위로 파일에 할당되는 것을 나타내고, 값 '00000001h'는 볼륨이 프레그먼트 단위로 파일에 할당되는 것을 나타낸다. 값 'FFFFFFFFh'는 파일이 계층 파일의 부모 파일임을 나타낸다. 그 밖의 다른 값들은 파일이 계층 파일의 자식 파일임을 나타낸다. 이 경우, 할당 형태 필드는 부모 파일의 첫 클러스터의 번호를 저장한다. 그러므로, 값 '00000000h' 또는 '00000001h'를 나타냄으로써 클러스터 단위의 볼륨 할당과 프레그먼트 단위의 볼륨 할당 사이에서 관리할 수 있다. 이는, 청구항 10에 기재된 계층 종류에 관한 정보에 해당한다. 값 'FFFFFFFFh'는 청구항 10에 기재된 파일 형태에 관한 정보에 해당한다.

파일의 선두 종료 위치를 나타내는 필드에는 4 바이트가 부여된다. 클러스터 단위로 파일에 할당되면, 파일의 선두 종료 위치는 도 3에 도시한 4)의 형태와 같고, 프레그먼트 단위로 할당되면, 파일의 선두 종료 위치는 도 4에 도시한 3)의 형태와 같다. 8 바이트가 부여되는 파일 크기 필드는 파일 크기를 저장한다.

도 5B는 클러스터 FAT의 일례를 설명하는 도면이다. 각 클러스터에 해당하는 FAT 엔트리는 영역 933 내지 964에 기록된다. 영역(933)(934)의 FAT 엔트리는 보존되고, 이 테이블이 FAT, 버전 등임을 나타내는 식별값은 이들 영역에 기록된다. 그러므로, 클러스터 FAT의 엔트리는 클러스터 번호 2부터 시작된다. 상기에서 설명한 클러스터 FAT와 후술될 프레그먼트 FAT는 청구항 10에 기재된 제 1 관리 정보에 대항한다.

도 5B에 도시한 클러스터 FAT에서, 이차 기록 장치(503)의 볼륨이 1 기가 바이트이고 클러스터 크기가 64 킬로바이트이면, 이차 기록 장치(503)의 볼륨은 16,384 개의 클러스터로 분할된다. 각 클러스터 엔트리는 32 비트(4 바이트)의 기록 영역을 가지고, 클러스터 엔트리들은 영역(935)의 클러스터 번호 2의 FAT 엔트리로부터 오름차순으로 배열된다. 더 자세히 설명하면, 클러스터 엔트리들은 영역 935, 936, 937, 938, 939, 940, 941 등의 순서로 배열되고, 따라서 클러스터 번호도 2, 3, 4, 5, 6, 7, 8 등으로 커진다. 클러스터 FAT는 크기가 약 64 킬로바이트이다. 파일의 경우에서와 같이, 클러스터 FAT는 볼륨의 클러스터 또는 프레그먼트에 저장되고, 클러스터 FAT의 위치는 시스템 영역에 나타나 있다.

도 5C는 프레그먼트 FAT의 일례를 설명하는 도면이다. 프레그먼트 FAT의 엔트리들은 클러스터 단위로 배열되고, 모든 클러스터에 대한 프레그먼트 FAT의 엔트리들을 포함한다. 도 5C에서, 각 프레그먼트의 크기가 512 바이트이고 각 클러스터가 128 프레그먼트로 분할되는 것으로 가정한다. 프레그먼트 FAT 엔트리들은 클러스터 FAT의 경우와 같이 클러스터 번호 '2'의 클러스터의 프레그먼트 번호 '0'을 갖는 프레그먼트에 해당하는 엔트리 965D에서 시작하여 오름차순으로 배열된다. 초기의 128 개의 엔트리들은 클러스터 번호 '2'의 클러스터에 속한다. 이와 같은 방식으로, 16,384개의 클러스터들에 대응되는 프레그먼트 엔트리

륨이 클러스터 번호가 증가하는 순서로 배열된다. 각 엔트리는 32 비트이므로, 프레그먼트 FAT의 전체 크기는 8 메가바이트이다. 파일의 경우와 마찬가지로, 프레그먼트 FAT는 볼륨 내의 클러스터나 프레그먼트에 저장되고, 프레그먼트 FAT의 위치는 시스템 영역으로부터 나타낸다. 프레그먼트 FAT와 클러스터 FAT는 청구항 2에 기재된 바와 같이 오름차순으로 배열된다.

도 5를 보면, 영역(901)의 FILE1.DAT와, 영역(905)의 FILE2.DAT와, 영역(909)의 FILE3.DAT는 볼륨이 클러스터 단위로 파일에 할당되는 것을 나타낸다. 이 경우, 할당 형태 필드를 구성하는 영역(902)(906)(910)의 값 '00000000h'로부터 클러스터 단위로 할당되고 있음을 알 수 있다.

영역(901)의 FILE1.DAT의 디렉토리 엔트리에 해당하는 영역(903)은 파일의 선두 종류 위치 즉, 파일을 구성하는 클러스터들 중에서 첫 번째 클러스터의 위치를 저장한다. 영역(903)의 상위 25 비트에 저장된 값 '000000h'는 파일의 첫 번째 클러스터의 번호가 '2'임을 나타낸다.

클러스터 FAT의 경우, 클러스터 번호 '2'에 대한 영역(935)의 FAT 엔트리의 상위 25 비트에 저장된 값 '0000004h'는 다음 클러스터의 클러스터 번호가 '4'임을 나타낸다. 클러스터 '4'의 다음 클러스터를 나타내는 클러스터 번호 '5'는 클러스터 번호 '4'에 대한 영역(937)에 저장된다. 마찬가지로, 클러스터 번호 '5'에 대한 영역(938)은 연속되는 클러스터의 번호 '8'을 저장하고, 클러스터 번호 '8'에 대한 영역(941)은 다음 클러스터의 번호 '9'를 저장한다. 그러므로, 영역 937, 938, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 953 및 954의 클러스터 FAT의 엔트리들을 이용하여 FAT 체인을 추적할 수 있다. 마지막으로 언급한 영역(955)에 저장된 엔트리의 상위 25 비트의 값은 '1FFFFFFh'이다. 이는, 클러스터가 파일을 구성하는 클러스터 중에서 가장 마지막 클러스터임을 나타낸다. 값 '1FFFFFFh'는 청구항 2에 기재한 파일의 끝을 나타내는 정보에 해당한다.

앞서 설명한 바와 같이, FILE1.DAT는 16개의 클러스터 즉, 클러스터 번호 2, 4, 5, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21 및 22의 클러스터들로 이루어져 있음을 알 수 있다. 디렉토리 엔트리의 영역(904)의 파일 크기 값은 1,032,091 바이트이고, 이 중 49,051 바이트만이 마지막 클러스터 22에 사용된다. 클러스터 단위로 구성되는 파일의 모든 클러스터 FAT 엔트리들은 도 3에 도시한 4)의 형태를 갖는다.

상기에서 설명한 방법으로 영역(905)의 FILE2.DAT와 영역(909)의 FILE3.DAT에 대한 클러스터 FAT를 추적할 수 있다. FILE2.DAT는 클러스터 '7'만으로 구성되고, FILE3.DAT는 클러스터 '23'과 클러스터 '6'으로 구성된 것을 알 수 있다.

영역(913)의 FILE4.DAT와 영역(917)의 FILE5.DAT는 프레그먼트 단위로 할당된 예이다. 이는, 할당 형태 필드 영역(914)(918)의 값이 '00000001h'인 것으로부터 알 수 있다.

영역(913)의 FILE4.DAT의 디렉토리 엔트리의 영역(915)은 파일을 구성하는 프레그먼트 중에서 첫 번째 프레그먼트의 위치를 저장한다. 영역(915)의 상위 25 비트의 값은 '0000012h'이고, 하위 7 비트의 값은 '00h'이다. 이는, 첫 번째 프레그먼트가 클러스터 번호 '18'의 클러스터의 프레그먼트 '0'임을 나타낸다.

도 5C를 보면, 첫 번째 프레그먼트의 프레그먼트 FAT의 엔트리에 대한 영역(969)이 클러스터 번호 '8'과 프레그먼트 번호 '0'을 나타낸다. 이는, 프레그먼트 FAT의 2176번째 FAT 엔트리(클러스터의 개수 (18-2) 프레그먼트 × 번호 128 + 1)를 나타낸다. 이 FAT 엔트리의 상위 25 비트 값은 '0000012h'이고 하위 7 비트 값은 '01h'이다. 그러므로, 다음 프레그먼트가 클러스터 '18'의 프레그먼트 '1'임을 알 수 있다.

클러스터 번호 '18'에 대응하는 다음 프레그먼트의 FAT 엔트리에 대한 영역(970)에서, 상위 25 비트는 '0000012h'를 나타내고 하위 7 비트는 '02h'를 나타낸다. 이는, 다음 프레그먼트가 클러스터 '18'의 프레그먼트임을 의미한다. 따라서, 하위 계층의 영역으로서 프레그먼트는 같은 수준의 다른 프레그먼트와의 연결을 나타내는 연결 정보와 사용 상태를 나타내는 정보와, 상위 계층의 영역인 클러스터와의 연결을 나타내는 연결 정보를 이용한다. 이것은 청구항 6에 기재한 특징에 해당한다.

마찬가지로, 클러스터 번호 '18'의 FAT 엔트리(971) 이후로 FAT 체인을 추적하여 클러스터 번호 '18'의 FAT 엔트리(976) 즉, 프레그먼트 번호 '2'에 이를 수 있다. 영역(976)의 FAT 엔트리의 상위 25 비트는 '1FFFFFFh'를 나타내어, 이 프레그먼트가 파일을 구성하는 프레그먼트 중에서 마지막 프레그먼트임을 의미한다.

앞서 설명한 동작에서, 파일이 네 개의 프레그먼트 즉, 클러스터 '18'의 프레그먼트 '0', 클러스터 '18'의 프레그먼트 '1', 클러스터 '18'의 프레그먼트 '2', 클러스터 '18'의 프레그먼트 '127'로 구성되어 있음을 알 수 있다. 영역(916)의 파일 크기의 값이 1581이므로, 영역(976)의 마지막 프레그먼트의 45 바이트가 파일에 사용됨을 알 수 있다(1581 바이트 - 512 바이트 × 3 = 45 바이트).

상기와 마찬가지로 프레그먼트 FAT 체인 또는 영역(917)의 FILE5.DAT를 추적할 수 있다. 이 경우, 프레그먼트 FAT 체인은 영역(919)의 첫 번째 프레그먼트 값으로부터 영역(977)의 엔트리를 지나서 영역(978)의 엔트리로 추적된다. 그러므로, 파일이 두 개의 프레그먼트 즉, 클러스터 '18'의 프레그먼트 '3'과 클러스터 '19'의 프레그먼트 '1'로 구성됨을 알 수 있다.

상기에 언급한 바와 같이, 클러스터 '18'과 클러스터 '19'는 프레그먼트 단위의 할당 형태에 따라 사용된다. 클러스터 '18'의 클러스터 FAT의 엔트리(951)의 상위 25 비트는 '0000000h'이고, 하위 7 비트는 '05h'이다. 이 엔트리는 도 3에 도시한 2)의 형태를 나타내고, 클러스터 '18'이 프레그먼트 단위로 사용되고 있음을 의미한다.

하위 7 비트의 값은 이 클러스터 내의 다섯 개의 프레그먼트들이 사용되고 있음을 나타낸다. 클러스터 '19'의 클러스터 FAT의 엔트리(952)의 상위 25 비트는 '0000000h'이고 하위 7 비트는 '01h'이다. 이는, 클러스터 19에 오직 한 개의 프레그먼트만이 사용되고 있음을 나타낸다. FILE1.DAT 내지 FILE5.DAT를 읽는 동작은 청구항 3에 기재된 특징의 실시예이다.

영역(921)의 FILE6.DAT의 파일의 할당 형태 필드(922)는 'FFFFFFFFh'를 나타내고, 이는, FILE6.DAT가 계층 파일의 부모 파일임을 의미한다. 영역(925)의 FILE7.DAT와 관련한 영역(927)에 저장된 파일의 선두 종

료 위치 값과 영역(926)에 저장된 할당 형태 값은 영역(929)의 FILE8.DAT의 영역(930)에 저장된 할당 형태 값과 마찬가지로 '00000018h'이다. 이는, 영역(923)에 나와 있는 FILE6.DAT의 영역(923)의 파일의 선두 종료 위치 값 '00000018h'와 일치한다. 이는, 영역(925)의 FILE7.DAT와 영역(929)의 FILE8.DAT가 모두 영역(921)의 FILE6.DAT의 자식 파일임을 의미한다.

계층 파일의 부모 파일의 FAT 체인을 추적하는 방법은 클러스터 단위 할당 형태의 일반 파일의 FAT 체인을 추적하는 방법과 동일하다. 영역(921)에 나와 있는 FILE6.DAT의 영역(923)에서 파일의 선두 종료 위치 값은 '00000018h'이다. 이는, 첫 번째 클러스터의 클러스터 번호가 '24'임을 의미한다.

클러스터 '24'에 대한 클러스터 FAT에서 엔트리(957)의 상위 25 비트는 다음 클러스터의 클러스터 번호가 '25'임을 나타내는 '0000019h' 값을 갖는다. 클러스터 '25'에 대한 클러스터 FAT에서 엔트리(958)의 상위 25 비트는 그 다음 클러스터의 클러스터 번호가 '26'임을 나타내는 '0000019h' 값을 갖는다. 클러스터 '26'에 대한 클러스터 FAT에서 엔트리(959)의 상위 25 비트는 이 클러스터가 파일을 구성하는 클러스터 중에서 최종 클러스터임을 나타내는 '1FFFFFFh' 값을 갖는다.

앞서 언급한 동작에서, 영역(921)의 FILE6.DAT가 세 개의 클러스터 즉, 클러스터 '24', 클러스터 '25' 및 클러스터 '26'으로 구성되는 것을 알 수 있다. 계층 파일의 부모 파일은 이들 클러스터 모두의 데이터를 사용하므로, 영역(924)의 파일 크기는 196,608 바이트에 이른다. 클러스터 FAT의 엔트리들(957)(958)(959)의 하위 7 비트 값은 각 클러스터에서 세 개의 프레그먼트, 세 개의 프레그먼트 및 두 개의 프레그먼트가 사용되고 있음을 나타내는 각각 '03h', '03h' 및 '02h'이다. 그러므로, 상위 계층의 클러스터는 연결 정보와 이용 상태 정보뿐만 아니라, 하위 계층의 영역인 프레그먼트의 이용 상태를 확인할 수 있다. 이 특징은 청구항 5에 해당한다.

영역(925)의 FILE7.DAT와 영역(929)의 FILE8.DAT는 자식 파일이다. 이들 자식 파일들에 부모 파일 영역(921)의 FILE6.DAT에 할당된 클러스터의 프레그먼트들을 이용하여 프레그먼트 단위로 할당이 이루어진다. 계층 파일의 자식 파일에 대한 프레그먼트 FAT를 추적하는 방법은 프레그먼트 단위로 볼륨이 할당되는 일반 파일의 경우와 동일하다.

예를 들면, 영역(925)의 FILE7.DAT의 경우, 프레그먼트 FAT 체인을 시작 종료 클러스터 번호 '24'와 프레그먼트 번호 '0'에서부터 시작하여 추적하면, 프레그먼트 FAT의 엔트리(981)와 엔트리(983)를 거쳐서 프레그먼트 FAT의 엔트리(991)에 이른다. 따라서, 파일이 세 개의 프레그먼트 즉, 클러스터 '24'의 프레그먼트 '0'; 클러스터 '24'의 프레그먼트 '2'; 및 클러스터 '26'의 프레그먼트 '2'로 이루어진 것을 알 수 있다.

영역(929)의 FILE8.DAT에 대해서도 동일한 추적 방법을 사용할 수 있다. 즉, 프레그먼트 FAT 체인을 시작 종료 클러스터 번호 '24'와 프레그먼트 번호 '1'에서부터 시작하여 추적하면, 프레그먼트 FAT의 엔트리(981)(985)(986)(987)를 거쳐서 프레그먼트 FAT의 엔트리(990)에 이른다. 따라서, 파일이 다섯 개의 프레그먼트 즉, 클러스터 '24'의 프레그먼트 '1'과, 클러스터 '25'의 프레그먼트 '0'과, 클러스터 '25'의 프레그먼트 '1'과, 클러스터 '25'의 프레그먼트 '2'와, 클러스터 '26'의 프레그먼트 '1'로 이루어진 것을 알 수 있다. FILE6.DAT 내지 FILE8.DAT를 읽기 위한 상기 동작은 청구항 4에 기재한 특징을 구현한다.

다음은 도 6 내지 32를 참조하여 본 발명에 따른 계층적 관리 파일 장치에 의해 수행되는 상기 동작들을 상세히 설명하기로 한다. 따라서, 다음 동작들 (1) 내지 (14)를 다음 순서로 설명하기로 한다. 이들 동작들 모두는 청구항 1에 기재된 소프트웨어 제어 수단의 제어 하에 수행된다.

(1) 파일 읽기

(2) 클러스터 단위로 읽기

(3) 프레그먼트 단위로 읽기

(4) 파일 쓰기

(5) 클러스터 단위로 쓰기

(6) 프레그먼트 단위로 쓰기

(7) 검색 범위로부터 새 클러스터 획득

(8) 프레그먼트 단위로 쓰기

(9) 새 프레그먼트 획득

(10) 검색 단위로부터 새 프레그먼트 획득

(11) 클러스터로부터 새 프레그먼트 획득

(12) 계층 파일의 새 프레그먼트 획득

(13) 새 파일 만들기

(14) 계층 파일의 새 자식 파일 만들기

(1) '파일 읽기' 동작의 설명

도 6 및 도 7은 '파일 읽기' 과정을 설명하는 흐름도를 보여 준다.

'파일 읽기' 동작은 파일의 경로명(PathName)을 설정하는 단계 S1000에서 시작된다. 흐름도에서 보이는 파라미터(Pathname)는 파일의 경로명을 나타낸다. 다른 파라미터를 나타내는 데에 이와 유사한 표현이 다른 흐름도에서도 사용된다. 단계 S1000에서는 또한 파일에서 읽기 시작 위치를 나타내는 바이트

오프셋(ByteOffset)과 읽을 바이트 수(ByteLength)를 설정한다. 바이트 오프셋은 파일의 선두 종료 위치에서부터 읽기가 시작되는 파일 내의 위치까지 바이트 수를 나타낸다. 그러므로, 선두 시작 위치는 항상 클러스터의 선두 종료 위치와 일치한다.

단계 S1001은 루트 디렉토리에 따라 (PathName)를 바탕으로 계층 디렉토리를 추적하여 파일의 디렉토리 엔트리를 검색한다. 단계 S1002는 파일이 존재하는지의 여부를 결정한다. 검색이 성공적으로 이루어지면 즉, 파일이 존재하면, 단계 S1003으로 진행하고, 그렇지 않으면 단계 S1004로 진행하여 오류 처리를 수행한다.

단계 S1003은 할당 형태(AllocType), 파일의 선두 종료 위치 즉, 클러스터나 프레그먼트, 위치, 및 파일 크기(FileSize)를 파일의 디렉토리 엔트리로부터 읽고 저장한다.

단계 S1005는 읽기 시작 위치와 읽기 종료 위치가 파일 크기를 초과하는지를 판단한다. 읽기 시작 위치와 읽기 종료 위치가 파일 크기보다 작으면, 단계 S1006으로 진행하고, 그렇지 않으면 단계 S1112로 진행하여 오류 처리를 수행한다.

단계 S1006은 할당 형태를 판단한다. 파일의 형태가 볼륨이 클러스터 단위로 할당되어 있는 형태이거나(AllocType == 0), 파일이 계층 파일의 부모 파일이면(AllocType == FFFFFFFFh), 다음에 설명할 '클러스터 단위로 읽기' 동작이 단계 S1007에서 수행된다.

다른 경우 즉, 파일의 형태가 볼륨이 프레그먼트 단위로 할당되어 있거나, 파일이 계층 파일의 자식 파일인 경우, 후술될 '프레그먼트 단위로 읽기' 동작이 단계 S1008에서 수행된다. 단계 S1007 또는 S1008의 처리가 끝나면, 단계 S1010으로 진행하여 '파일 읽기' 동작을 종료한다.

(2) '클러스터 단위로 읽기' 동작의 설명

도 8 및 도 9는 '클러스터 단위로 읽기' 과정을 설명하는 흐름도를 보여 준다. 이 동작 모드에서, 클러스터 FAT 엔트리의 체인을 추적하면서 읽기가 수행된다. 클러스터 단위로 읽기 동작은 파일의 선두 종료 위치(Location), 파일 내 읽기 시작 위치의 바이트 오프셋(ByteOffset), 그리고 읽을 바이트 수(ByteLength)를 설정하는 단계 S1101에서 시작한다. 파일의 선두 종료 위치와 읽기 시작 위치와 읽을 바이트 수는 디렉토리 엔트리에서 얻는다.

단계 S11-2는 단계 S1102의 블록에서 설명하고 있는 계산을 통하여 시작 클러스터 어드레스(ClusterAddress), 파일 내 읽기 위치의 클러스터 오프셋(ClusterStart), 그리고 읽기 종료 위치의 클러스터 오프셋(ClusterEnd)을 ??정한다. 단계 S1102는 클러스터 오프셋(ClusterOffset)의 변수를 0으로 초기화한다. 클러스터 오프셋의 변수는 정수이고 각 클러스터를 읽을 때마다 '1'씩 증가한다. 파일의 시작 종료 위치(Location)는 도 3에 나타낸 4)의 형태로 기록되었다. 시작 클러스터의 어드레스(ClusterAddress)는 파일의 시작 종료 위치(Location)의 상위 25 비트에 따라 결정된다. 클러스터 크기(ClusterSize)는 볼륨 내에 고정된 값으로서, 시스템 영역으로부터 읽어들일 수 있다. 상기에 언급한 클러스터 어드레스는 상기 클러스터 번호에 해당한다.

단계 S1103은 읽기 시작 위치의 클러스터 오프셋 변수(ClusterOffset)가 읽기 시작 위치의 클러스터 오프셋(ClusterStart)보다 작은지를 판단한다. 그렇다면, 읽어들일 데이터가 이 클러스터 내에 존재하는 것이므로 단계 S1104로 진행하여 읽기 처리로 수행한다. 이와 반대로, 클러스터 오프셋 변수(ClusterOffset)가 읽기 시작 위치(ClusterStart)의 클러스터 오프셋보다 작지 않으면, 단계 S1104를 생략하고 단계 S1105로 진행한다.

그러므로, 단계 S1104가 실질적으로 클러스터 데이터의 읽기를 수행하는 것이다. 시작 클러스터(ClusterStart)로부터 데이터를 읽을 때, 파일 내 읽기 위치의 바이트 오프셋(ByteOffset)이 클러스터 크기의 배수가 아니면, 시작 클러스터 상의 읽기 위치의 바이트 오프셋이 ByteOffset%ClusterSize로 설정된다. 여기서, %는 'ByteOffset'과 ClusterSize의 모듈로 계산 결과를 나타내고, 다음 설명에서와 동일한 의미로 사용된다. 마찬가지로, 마지막 클러스터(ClusterEnd)를 읽을 때, (ByteOffset + ByteLength)가 클러스터 크기의 배수가 아니면, 마지막 클러스터로부터 읽어들일 바이트 수가 (ByteOffset+ByteLength)%ClusterSize로 설정된다. 다른 경우, 클러스터의 모든 데이터가 읽힌다.

단계 S1105는 현재의 클러스터 오프셋(ClusterOffset)이 종료 위치(ClusterEnd)의 클러스터 오프셋과 일치하는지를 판단한다. 그렇다면, 단계 S1106으로 진행하여 '클러스터 단위로 읽기' 동작을 종료한다. 그러나, 현재의 클러스터 오프셋이 종료 위치의 클러스터 오프셋과 일치하지 않으면, 단계 S1107로 진행하여 현재의 클러스터 오프셋 값(ClusterOffset)을 '1'만큼 증가시킨다. 그런 다음, 단계 S1108로 진행하여 현재 (증가된) 클러스터의 FAT 엔트리(Entry)의 값들을 클러스터 FAT로부터 읽어들인다.

단계 S1109는 읽기 엔트리(Entry)가 현재 클러스터가 파일을 구성하는 클러스터 중에서 마지막 클러스터임을 나타내는지를 판단한다. 도 6에 나타낸 흐름도에서 단계 S1005는 파일 크기를 판단하였으므로 현재 클러스터는 마지막 클러스터가 될 수 없다. 도 3에 나타낸 4)의 형태와 5)의 형태에서 상위 25 비트가 '1FFFFFFh'의 값을 가지면, 단계 1110으로 진행하여 오류 처리를 수행한다.

단계 S1109에서 현재 클러스터가 마지막 클러스터임이 판단되면, 다음 클러스터 어드레스의 상위 25 비트(Entry.High25Bit)는 현재 클러스터의 클러스터 어드레스(ClusterAddress)를 대신한다. 그런 다음, 단계 S1103으로 되돌아가서 단계 S1105가 현재 클러스터 오프셋(ClusterOffset)이 종료 위치의 클러스터 오프셋(ClusterEnd)과 일치한다고 판단할 때까지 상기 동작을 반복하여 필요한 모든 데이터를 읽어들인다.

(3) '프레그먼트 단위로 읽기' 동작의 설명

도 10 및 도 11은 '프레그먼트 단위로 읽기' 과정을 설명하는 흐름도를 보여 준다. 이 과정은 프레그먼트 FAT의 FAT 엔트리의 체인을 추적하면서 수행된다. 단계 S1208이 프레그먼트 FAT를 이용하고, 단계 S1203과 S1211에서 사용된 엔트리들이 도 4에 나타낸 3)의 형태를 갖는 것을 제외하고는, 프레그먼트 단위로 읽기 동작은 도 8 및 도 9에서 앞서 설명한 '클러스터 단위로 읽기' 동작과 같다. 다음 방법에서와 같이

프레그먼트 어드레스는 상기 프레그먼트 번호에 해당한다.

(4) '파일 쓰기' 동작의 설명

도 12 및 도 13은 '파일 쓰기' 과정을 설명하는 흐름도를 보여 준다. 파일의 쓰기 동작은 도 6 및 도 7을 참조하여 앞서 설명한 '파일 읽기' 동작과 기본적으로 동일하게 수행된다. 여기에는, 단계 1310에서 시작하여 후술될 '클러스터 단위로 쓰기' 동작을 수행하는 것과, 단계 1311에서 시작하여 후술하는 '프레그먼트 단위로 쓰기' 동작을 수행하는 두 가지 방법이 있다. '파일 쓰기' 동작은 파일 크기를 결정하지 않고 수행된다. 쓰기 동작이 진행되면서, 새 클러스터 또는 새 프레그먼트가 할당되어 파일의 크기(FileSize)가 증가된다. 이 경우, 디렉토리 엔트리가 단계 S1312에서 갱신된다. 제 2 관리 정보를 구성하는 디렉토리 엔트리의 갱신은 청구항 11에 기재된 특징을 구현한다.

기존의 파일로 쓰기가 가능하다. 그러나, 디렉토리 엔트리에 파일이 존재하지 않으면, 시스템에 따라 새 파일을 만들 수 있다. 그러므로, 단계 1303에서 수행된 검색을 통하여 파일이 발견되지 않으면, 단계 S1306으로 진행하여 '새 파일 만들기' 동작이 수행된다.

(5) '클러스터 단위로 쓰기' 동작의 설명

도 14 및 도 15는 '클러스터 단위로 쓰기' 과정을 설명하는 흐름도를 보여 준다. '클러스터 단위로 쓰기' 동작은 도 8 및 도 9를 참조하여 앞서 설명한 '클러스터 단위로 읽기' 동작과 기본적으로 동일하다. 단계 1401에서 데이터를 클러스터로 쓰는 동작이 수행된다. 읽기 동작의 경우에서와 같이, 시작 클러스터에 바이트 오프셋이 존재하지만 데이터는 마지막 클러스터를 완전히 채울 수 없다. 쓰기 동작에서는, 같은 클러스터에서 다른 데이터를 바꾸지 않고 다시 쓸 데이터만을 갱신해야 한다.

파일 크기가 기록 데이터를 수용할 정도로 크지 않으면, 새 클러스터를 할당해야 한다. 단계 1409에서 현재의 클러스터가 파일을 구성하는 클러스터 중에서 마지막 클러스터라고 판단하면, 단계 1410으로 진행하여 '새 클러스터 획득' 동작을 수행한다.

흐름도에 나타낸 과정에서, 획득이 필요할 때마다 한 개의 클러스터가 획득된다. 그러나, 이에 한정되는 것이 아니며, 기록할 전체 데이터를 수용하기 위한 새로 획득할 클러스터 수를 단계 S1402에서 미리 결정하여 부가로 사용되는 모든 클러스터를 쓰기 동작을 시작하기 전에 획득하는 것과 같이 과정을 변형시킬 수 있다. 이 경우, 데이터를 연속 클러스터 내에 순차적으로 기록하기 위하여, 후술하는 '새 클러스터 획득' 동작에서 사용하지 않은 클러스터를 검색하여 연속적인 미사용 클러스터의 획득 개수를 알아낸다. 이것은 청구항 9에 기재한 특징을 구현한다.

(6) '새 클러스터 획득' 동작의 설명

도 16은 '새 클러스터 획득' 과정을 설명하는 흐름도를 보여 준다. 이 동작에서 중요한 것은 사용하지 않은 클러스터를 찾아내는 일이다. 도 16의 흐름도에서, 사용하지 않은 클러스터의 탐색이 서로 다른 검색 범위에서 단계 S1502와 단계 1504에서 두 번 수행된다. 이를 자세히 설명하면, 단계 S1502에서는 볼륨 내의 현재 클러스터에서 마지막 클러스터까지 검색이 이루어지고, 단계 S1504에서는 볼륨의 첫 번째 클러스터에서 현재 클러스터까지 검색이 이루어진다. 두 단계를 수행하였어도 새 클러스터 획득이 실패하면, 단계 1506으로 진행하여 오류 처리를 수행한다.

새 클러스터 획득 동작의 기본 방안은 탐색 시간을 최소화하기 위하여 현재 클러스터에 가장 가까운 위치에서 사용하지 않은 클러스터를 찾아내는 것이다. 이를 위하여, 각 단계 S1502와 S1504에서 검색 범위가 여러 영역으로 세분되고, 현재 클러스터에서 가장 가까운 영역으로부터 시작하여 현재 클러스터에서 전후 방향으로 검색을 수행하는 것이다.

단락 (5)에서 설명한 바와 같이, 새로 필요한 클러스터의 전체 수를 알고 있는 경우, 사용하지 않은 클러스터를 나중에 사용하도록 찾고, 떨어진 위치에서 클러스터를 찾는 데에 걸리는 전체 탐색 시간보다 이들 클러스터의 탐색 시간의 합이 최소가 되도록 하는 것이 바람직하다. 이는, 클러스터의 이용 상태를 판단하여 검색 범위의 각 영역에 속하는 사용하지 않은 클러스터 수를 나타내는 정보를 얻으면 된다. 이 것은 청구항 1과 청구항 5에 기재한 특징에 해당한다.

보다 많은 수의 클러스터를 탐색하려면 검색 처리 시간이 길어진다. 본 발명에 따르면, 파일 장치(510)는 각 클러스터의 크기가 비교적 크고 클러스터의 수가 감소되는 계층적 FAT를 이용한다. 그러므로, 사용하지 않은 클러스터를 효율적으로 검색할 수 있다.

(7) '검색 범위로부터 새 클러스터 획득' 동작의 설명

도 17 및 도 18은 '검색 범위로부터 새 클러스터 획득' 과정을 설명하는 흐름도를 보여 준다.

이 과정은 현재 클러스터 어드레스(ClusterAddress)를 검색이 시작되는 클러스터의 클러스터 어드레스(FindCluster)로 설정하고 파일의 마지막 클러스터의 어드레스를 검색이 끝나는 클러스터의 클러스터 어드레스(FindEnd)로 설정하는 단계 S1601에서 시작한다.

단계 S1602는 클러스터가 검색 범위 내에 있는지를 판단한다. 파라미터(FindCluster)는 보다 큰 정수로 변화되어 검색시에 클러스터의 어드레스로 사용된다. FindCluster의 값이 마지막 클러스터(FindEnd) 보다 크지 않음을 확인하여 클러스터가 검색 범위 이내에 있는지를 판단한다. 클러스터가 검색 범위 이내에 있으면, 단계 S1603으로 진행하고, 그렇지 않으면 단계 S1611로 진행한다. 단계 S1611에서, 클러스터를 찾지 못하면서 검색이 종료된다.

단계 S1603은 클러스터 FAT로부터 클러스터 어드레스(FindCluster)의 엔트리 값(Entry)을 읽는다. 그런 다음, 단계 S1604는 클러스터가 사용하지 않은 클러스터인지를 판단한다. 클러스터가 도 30에서 도시한 1)의 형태를 가지고 상위 25 비트가 '00000000h' 값을 가지면, 클러스터는 사용하지 않은 클러스터로 판단된다. 이 경우, 단계 S1605로 진행하고, 검색을 위한 현재 클러스터 어드레스는 '1'만큼 증가한다. 그런 다음, 단계 S1602로 되돌아가서 다음 클러스터를 검사한다. 이 동작은 사용하지 않은 클러스터가 검색 범

위 이내에서 발견될 때까지 반복된다.

단계 S1604에서 사용하지 않은 클러스터가 발견되면, 단계 S1606으로 진행하여 FAT 엔트리의 새로운 값을 설정한다. FAT 엔트리의 새로운 값은 항상 마지막 클러스터를 나타내는 FAT 엔트리로 설정된다. 그러면, FAT 엔트리는 도 3에 도시한 4)의 형태를 갖게 되고 상위 25 비트는 'FFFFFFh'의 값을 갖는다. 단계 S1607에서, 상기 FAT 엔트리 값(Entry)이 검색된 사용하지 않은 클러스터의 FAT 엔트리(FindCluster)로 기록되어 FAT 엔트리가 갱신된다.

사용하지 않은 클러스터를 찾은 후에 수행되는 제 1 관리 정보로서 클러스터 FAT 엔트리의 갱신과, 후술되는 사용하지 않은 프레그먼트를 찾은 후에 수행되는 프레그먼트 FAT 엔트리의 갱신은 청구항 10에 기재된 특징을 구현한다.

단계 S1608에서, 현재의 마지막 클러스터의 FAT 엔트리가 검색을 통하여 찾아낸 사용하지 않은 클러스터를 나타내도록 설정된다. 그러면, FAT 엔트리는 도 3에 도시한 4)의 형태를 갖게 되고, FindCluster는 이 형태의 상위 25 비트 값으로 설정된다. 단계 S1609에서, 상기 엔트리(Entry)가 클러스터 어드레스(ClusterAddress)에서 FAT 엔트리 내에 기록되어 FAT 엔트리가 갱신된다. 그러나, 새 파일을 만드는 경우, 첫 번째 클러스터가 '새 클러스터 획득' 과정을 통하여 획득된다. 이 경우, 선두 종료 위치는 현재 FAT 값을 갱신할 필요 없이 디렉토리 엔트리에 의해 표시되므로, 단계 S1608과 S1609의 동작 수행이 필요하지 않게 된다.

FAT 체인은 단계 S1606 내지 S1609의 동작의 결과로 갱신된다. 그런 다음, 단계 S1610은 검색 결과로 찾아낸 클러스터(FindCluster)를 현재 마지막 클러스터가 되게 설정하도록 현재의 클러스터 어드레스(ClusterAddress)를 치환하고, 단계 S1611에서 '검색 범위로부터 새 클러스터 획득' 동작을 종료한다.

(8) '프레그먼트 단위로 쓰기' 동작의 설명

도 19 및 도 20은 '프레그먼트 단위로 쓰기' 과정을 설명하는 흐름도를 보여 준다. '프레그먼트 단위로 쓰기' 과정은 도 14 및 도 15를 참조하여 앞서 설명한 '클러스터 단위로 쓰기' 과정과 기본적으로 동일하다. 디렉토리 엔트리로 표시되는 파일의 선두 종료 위치(클러스터)로부터 클러스터 FAT를 사용하는 '클러스터 단위로 쓰기' 동작과 대조적으로, '프레그먼트 단위로 쓰기' 동작은 프레그먼트 FAT를 사용한다. 쓰기의 결과로 파일 크기가 증가하면, 아래에 설명되는 '새 프레그먼트 획득' 과정을 단계 S1710에서 수행한다.

(9) '새 프레그먼트 획득' 동작의 설명

도 21은 '새 프레그먼트 획득' 과정을 설명하는 흐름도를 보여 준다. 도 16을 참조하여 앞서 설명한 '새 클러스터 획득' 과정과 마찬가지로, '새 프레그먼트 획득' 과정은 분할된 검색 범위에 거쳐 검색을 수행한다. 단계 S1802는 대상 파일이 계층 파일의 자식 파일인지를 판단한다. 대상 파일이 계층 파일의 자식 파일이면, 단계 S1804로 진행하여 후술되는 방법에 따라 '계층 파일의 새 프레그먼트 획득' 동작을 수행한다.

일반 파일의 경우, 단계 S1803으로 진행하여 '새 프레그먼트 획득' 동작을 수행한다. 클러스터 FAT를 참조하여 각 클러스터 내 프레그먼트의 이용 상태를 검사한 후, 해당 클러스터의 범위 내에서 사용하지 않은 프레그먼트들을 찾아낸다. 그럼 다음, 후술할 '검색 범위로부터 새 프레그먼트 획득' 과정을 수행하여, 현재 프레그먼트 어드레스(FragmentAddress)의 상위 25 비트 값에 따라 클러스터 어드레스를 결정한다. 그리고 나서, 단계 S1803은 볼륨의 현재 클러스터에서 마지막 클러스터까지의 범위에 거쳐 검색을 수행한다. 검색이 실패하면, 단계 S1806으로 진행하여 볼륨의 첫 번째 클러스터에서 현재 클러스터까지의 범위에서 검색을 수행한다. 새 프레그먼트가 획득되지 않으면, 단계 S1808에서 오류 처리를 수행한다. 새 프레그먼트가 획득되면, 단계 S1809에서 '새 프레그먼트 획득' 과정을 종료한다.

(10) '검색 범위로부터 새 프레그먼트 획득' 동작의 설명

도 22 및 도 23은 '검색 범위로부터 새 프레그먼트 획득' 과정을 설명하는 흐름도를 보여 준다. 이 과정은 단계 S1901에서 시작하여, 단락 (9)의 '새 프레그먼트 획득' 과정에서 설명한 바와 같이 시작 클러스터(FindCluster), 검색 범위의 종료 클러스터(FindEnd), 그리고 대상 파일의 마지막 프레그먼트의 어드레스(FragmentAddress)와 같은 파라미터들을 설정하여 새 프레그먼트를 찾기 위한 검색 범위를 설정한다. 목적 클러스터(FindCluster)는 정수로 주어지는 변수이고, 검색되는 클러스터를 나타낸다. 새 프레그먼트의 검색은 클러스터 FAT의 검사와 프레그먼트 FAT의 검사의 두 단계로 수행된다. 그러므로, 상위 계층에 있는 클러스터 상에서 검색이 먼저 수행된 후에, 하위 계층의 프레그먼트 상에서 검색이 이루어진다. 이것은 청구항 12에 기재된 특징을 구현한다.

단계 S1902는 검색시에 검사되고 있는 클러스터가 검색 범위 이내에 있는지를 판단한다. 'FindCluster'가 'FindEnd'보다 크지 않으면, 검사되고 있는 클러스터가 검색 범위 이내에 있는 것으로 판단하고 단계 S1903으로 진행하고, 그렇지 않으면 단계 S1909로 진행하여 새 프레그먼트 획득이 실패하였으므로 과정을 종료한다.

검사되고 있는 클러스터가 검색 범위 이내에 있지 않으면, 단계 S1903으로 진행하여 검사되고 있는 클러스터(FindCluster)에 해당하는 클러스터 FAT의 엔트리 값(ClusterEntry)을 읽는다. 단계 S1904에서, 엔트리 값이 도 3에 도시한 1)의 형태를 갖고 '00000000h'의 값을 나타내면, 클러스터는 사용하지 않은 클러스터로 판단되고 eksrP S1907로 진행하여 'B)사용하지 않은 클러스터로부터 새 프레그먼트 획득' 과정을 수행한다.

단계 S1905에서 엔트리 값이 도 3에 도시한 2)의 형태를 갖고 '00000100h'의 값을 나타내는 것으로 판단되면, 클러스터가 사용하지 않은 프레그먼트를 갖는 것으로 간주한다. 이 경우, 단계 S1908로 진행하여 후술하는 'A) 사용중인 클러스터로부터 새 프레그먼트 획득' 과정을 수행한다. 단계 S1904와 S1905에서 긍정적인 대답을 얻지 못하면, 단계 S1906으로 진행하여 검색시에 클러스터의 클러스터

어드레스(ClusterAddress)를 '1'만큼 증가시킨다. 그런 다음, 단계 S1092로 되돌아가서 검색 범위 내에서 단계 S1902 내지 S1905의 동작을 반복한다.

획득되는 프레그먼트가 가능한 한 동일한 클러스터에 포함되도록 하기 위해서, 단계 S1905는 각 클러스터에서 사용중인 프레그먼트의 수를 나타내는 (ClusterEntry)의 하위 7 비트에 따라서 쓰기 동작을 위하여 획득될 다수의 프레그먼트를 수용하기에 충분히 많은 수의 미사용 프레그먼트들을 갖는 클러스터를 선택하는 것이 바람직하다. 보다 많은 수의 미사용 프레그먼트들을 갖는 클러스터를 선택하는 것은 청구항 5에 기재된 특징을 구현한다.

(11) '클러스터로부터 새 프레그먼트 획득' 동작의 설명

도 24, 도 25 및 도 16은 '클러스터로부터 새 프레그먼트 획득' 과정을 설명하기 위한 흐름도를 보여 준다.

이 과정은 'A) 사용중인 클러스터로부터 새 프레그먼트 획득'의 경우와 'B) 사용하지 않은 클러스터로부터 새 프레그먼트 획득'의 경우에 따라 두 가지로 나누어진다.

'A) 사용중인 클러스터로부터 새 프레그먼트 획득' 과정은 현재 프레그먼트 어드레스(FragmentAddress)와, 검색 클러스터의 클러스터 어드레스(FindCluster)와, 그리고 검색 클러스터의 FAT 엔트리 값(ClusterEntry)을 설정하는 단계 S2001로 시작된다. 단계 S2002는 검색 시작 프레그먼트 어드레스와 검색 종료 프레그먼트 어드레스를 설정하여 검색 범위를 정의하며, 범위가 목적 클러스터 이내에 해당하도록 한다. 클러스터의 선두 종료 프레그먼트의 어드레스는 상위 25 비트에 클러스터 값을 설정하고 하위 7 비트에 값 '0'을 설정하여 정한다. 이렇게 하여, 종료 프레그먼트의 어드레스는 시작 프레그먼트의 어드레스에 127을 더하여 결정한다. 검색이 수행되고 있는 클러스터를 나타내는 파라미터 'FindCluster'는 정수로 주어진 변수이고, 검색이 다른 클러스터로 옮겨질 때마다 변한다.

단계 S2003은 검색되고 있는 프레그먼트가 검색 범위 이내에 있는지를 판단한다. 아직 종료 클러스터에 이르지 않았으면, 과정이 단계 S2004로 진행하고, 그렇지 않으면 단계 S2005로 진행한다. 그러나, 클러스터 FAT의 엔트리 값이 적어도 한 개의 사용하지 않은 프레그먼트가 존재함을 나타내고 있기 때문에 오류가 발생하지 않는다.

[0133]

단계 S2004는 프레그먼트 FAT로부터 검색되고 있는 프레그먼트의 엔트리 값(FindFragment)을 읽는다. 단계 S2006은 이 프레그먼트가 사용하지 않은 프레그먼트인지를 검사한다. 엔트리 값이 도 5에 도시한 1)의 형태 즉, '0000000h'이면, 사용하지 않은 클러스터가 검색되고 단계 S2008로 진행한다.

[0134]

단계 S2006에서 프레그먼트가 사용하지 않은 프레그먼트가 아닌 것으로 판단하면, 검색 하에 프레그먼트(FindFragment)를 '1'만큼 증가시키고 단계 S2003으로 진행한다. 이렇게 하여, 검색의 전체 범위에 거쳐 검색이 끝날 때까지 단계 2003 내지 S2007을 반복한다.

사용하지 않은 프레그먼트를 찾지 못하면, 프레그먼트가 속하는 클러스터의 FAT 엔트리 값(ClusterFind)을 '1'만큼 증가시켜 사용중인 프레그먼트의 수를 증가시킨다. 그런 다음, 단계 S2011로 진행하여 클러스터 어드레스의 엔트리 값(FindCluster)을 'ClusterEntry'로 갱신한다.

단계 S2012는 프레그먼트 FAT 엔트리 값(FragmentEntry)을 도 4에 도시한 3)의 형태 즉, 'FFFFFF80h'로 설정하여 엔트리 값이 파일의 마지막 프레그먼트를 나타내도록 한다. 단계 S2013은 검색된 미사용 프레그먼트의 프레그먼트 어드레스(FindFragment)에서 프레그먼트 FAT 엔트리에 값(FragmentEntry)을 기록하여 프레그먼트 FAT 엔트리를 갱신한다.

단계 S2014는 프레그먼트 FAT의 엔트리 값(FragmentEntry)을 설정하여 검색된 미사용 프레그먼트의 어드레스(FindFragment)를 나타내도록 한다. 단계 S2015는 대상 파일의 프레그먼트 어드레스(FragmentAddress)의 프레그먼트 FAT의 엔트리에 프레그먼트 엔트리 값(FragmentEntry)을 기록한다. 그러나, 프레그먼트 어드레스(FragmentAddress)가 0이면 프레그먼트 FAT의 엔트리 값이 디렉토리 엔트리에 의해 지정되므로 갱신 동작이 수행되지 않는다.

파일에 해당하는 프레그먼트 FAT 체인인 단계 S2012 내지 S2015를 통하여 갱신된다. 단계 S2016은 검색된 미사용 프레그먼트를 파일의 마지막 프레그먼트의 프레그먼트 어드레스로 설정하고, 단계 S2017에서 과정을 종료한다.

'B) 사용하지 않은 클러스터로부터 새 프레그먼트 획득' 과정은 단계 S2009에서 시작한다. 클러스터가 사용하지 않은 클러스터이면, 이 클러스터의 모든 프레그먼트도 사용하지 않은 것이다. 이 경우, 이 클러스터의 첫 번째 프레그먼트가 새 프레그먼트로 정해진다. 단계 S2010은 도 30에 도시한 1)의 형태에 따라 값 '1'을 프레그먼트의 사용을 나타내는 하위 7 비트로 바꿔서, 사용되고 있지 않는 것으로 등록된 클러스터를 사용중인 클러스터로 바꾼다. 이와 동시에, 검색된 미사용 프레그먼트를 포함하는 클러스터의 클러스터 어드레스는 상위 25 비트에 설정되고, '0'은 하위 7 비트에 설정되어, 사용하지 않은 프레그먼트의 프레그먼트 어드레스를 형성한다.

그런 다음, 단계 S2011로 진행하여 'A) 사용중인 클러스터로부터 새 프레그먼트 획득' 과정과 같은 방법으로 클러스터 FAT의 엔트리를 갱신하고, 단계 S2013에서 프레그먼트 FAT의 엔트리를 검색된 프레그먼트에 따라 갱신한다. 단계 S2015에서 파일의 현재 프레그먼트 어드레스가 갱신되고, 단계 S2017에서 과정이 종료된다.

(12) '계층 파일의 새 프레그먼트 획득' 동작의 설명

도 27 및 도 28은 '계층 파일의 새 프레그먼트 획득' 과정을 설명하기 위한 흐름도를 보여 준다. 부모 파일의 자식 파일인 경우, 부모 파일에 할당된 클러스터들로부터 사용될 새 프레그먼트를 검색한다. 클러스

터를 중에서 하나의 클러스터가 사용하지 않은 프레그먼트를 가지고 있으면, 새 클러스터를 부모 파일에 할당하고, 새로 할당된 클러스터를 통하여 사용한 프레그먼트를 검색한다. 이 과정은 현재 프레그먼트(FragmentAddress)와 할당 형태(AllocType)를 설정하는 단계 S2101에서 시작한다. 이 단락에서 설명한 특징은 새 영역의 검색 및 할당을 구현한다.

대상 파일이 계층 파일의 자식 파일이면, 부모 파일의 시작 어드레스가 디렉토리 엔트리의 할당 형태(AllocType)에 설정된 것이므로, 단계 S2102에서 이 어드레스를 검색이 수행되는 클러스터의 어드레스(ClusterAddress)로 설정한다.

단계 S2103에서, 클러스터 FAT로부터 클러스터 어드레스(ClusterAddress)의 엔트리 값(ClusterEntry)을 읽는다. 단계 S2104는 엔트리 값이 도 4에 도시한 3)의 형태를 갖는지와 클러스터가 사용하지 않은 프레그먼트를 포함하는지를 판단한다. 하위 7 비트의 값이 '0'이 아니면, 단계 S2107로 진행하여 앞서 설명한 'A) 사용중인 클러스터로부터 새 프레그먼트 획득' 과정을 수행한다.

단계 S2104에서 사용하지 않은 클러스터를 찾지 못하면, 단계 S2105로 진행하여 클러스터가 마지막 클러스터인지를 검사한다. 엔트리 값이 도 3에 도시한 5)의 형태를 가지고 클러스터 번호가 '1FFFFFFh'를 나타내면, 단계 S2105로 진행하여 검사중인 클러스터가 마지막 클러스터인지를 판단한다. 그리고, 단계 S2108에서 앞서 설명한 '새 클러스터 획득' 과정을 수행하여 부모 파일의 새 클러스터를 획득한다.

단계 S2105에서 클러스터가 마지막 클러스터가 아니라고 판단되면, 단계 S2106으로 진행하여 엔트리 값의 상위 25 비트를 다음 클러스터의 클러스터 어드레스로 설정한다. 그런 다음, 단계 S2105로 되돌아가서 단계 S2104 내지 S2106의 동작을 반복한다. 단계 S2107에서 'A) 사용중인 클러스터로부터 새 프레그먼트 획득' 동작을 종료하면, 단계 S2111로 진행하여 과정을 종료한다.

단계 S2108은 앞서 설명한 '(6) 새 클러스터 획득' 동작을 수행한다. 새 클러스터의 획득 동작에서는 부모 파일의 디렉토리 엔트리를 갱신해야 한다. 그러므로, 단계 S2109에서, 부모 파일의 시작 어드레스(AllocType)에 따라 디렉토리 내의 부모 파일이 검색되고, 부모 파일의 다른 파라미터들이 갱신된다. 단계 S2110은 새로 할당된 클러스터 상에서 'B) 사용하지 않은 클러스터로부터 새 프레그먼트 획득' 동작을 수행하고, 단계 S2111에서 과정을 종료한다.

(13) '새 파일 만들기' 동작의 설명

도 29 및 도 30은 '새 파일 만들기' 과정을 설명하기 위한 흐름도를 보여 준다. 새 파일 만들기는 도 12 및 도 13을 참조하여 앞서 설명한 '파일 쓰기' 동작에서 소정의 파일을 찾지 못하였을 때와 새 파일을 만들고자 할 때에 수행된다. 그러나, 부모 파일이 존재할 때에만 자식 파일을 만들 수 있다. 새로운 자식 파일을 만드는 과정은 도 31 및 도 32를 참조하여 따로 설명하기로 한다.

'새 파일 만들기' 동작은 만들 파일의 경로명(PathName)과 할당 형태(AllocType)를 지정하는 단계 S2201에서 시작한다. 단계 S2202는 경로명 'PathName'을 갖는 파일의 디렉토리 엔트리를 마련한다. 단계 S2203은 디렉토리 엔트리가 만들어졌는지를 판단한다. 디렉토리 엔트리가 만들어졌으면, 단계 S2204로 진행하고, 그렇지 않으면 단계 S2205로 진행하여 오류 처리를 수행한다.

단계 S2204는 만들 파일이 계층 파일의 부모 파일인지를 판단한다. 파라미터 'AllocType'이 'FFFFFFFFh'의 값을 가지면, 단계 S2211로 진행하여 부모 파일을 만든다. 단계 S2206에서, 새 파일을 클러스터 단위로 만들 것인지를 결정한다. 파라미터 'AllocType'이 '0'의 값을 가지면, 단계 S2209로 진행하여 클러스터 단위로 파일을 만들고, 그렇지 않으면, 프레그먼트 단위로 파일을 만든다. 앞서 설명한 '(6) 새 클러스터 획득' 과정에 따라 단계 S2210에서 부모 파일을 만든다. 파라미터 'ClusterAddress'는 '0'으로 설정되어 파일이 클러스터를 가지고 있지 않음을 나타낸다.

단계 S2212는 디렉토리 엔트리의 값을 설정한다. 계층 파일의 부모 파일을 나타내는 값 'FFFFFFFFh'는 할당 형태 'AllocType'에 설정되고, 단계 S2210에서 새로 획득된 클러스터 어드레스(ClusterAddress)는 파일의 선두 종료 위치(Location)의 상위 25 비트에 설정된다. 파일 크기는 '0'으로 정해진다.

클러스터 단위로 파일 만들기는 단계 S2209에서 수행되는 '(6) 새 클러스터 획득' 과정을 이용한다. 단계 S2210에서 디렉토리 엔트리가 설정된다. 이 경우, 클러스터 단위로 할당되는 것을 나타내는 값 '0'이 할당 형태 'AllocType'에 설정된다.

프레그먼트 단위로 파일 만들기는 단계 S2207에서 수행되는 '(9) 새 프레그먼트 획득' 과정을 이용한다. 단계 S2208에서 디렉토리 엔트리가 설정된다. 이 경우, 프레그먼트 단위로 할당되는 것을 나타내는 값 '1'이 할당 형태 'AllocType'에 설정된다. 이와 동시에, 단계 S2207에서 획득된 새 프레그먼트의 어드레스가 파일의 선두 종료 위치(Location)로서 정해진다. 따라서, 디렉토리 엔트리의 설정은 파일이 클러스터 단위로 형성되는 경우와 프레그먼트 단위로 형성되는 경우의 두 가지로 각각 수행된다. 각각의 경우에서, 단계 S2213으로 진행하여 디렉토리 엔트리를 갱신하고 단계 S2214에서 과정이 종료된다.

(14) '계층 파일의 새로운 자식 파일 만들기' 동작의 설명

도 31 및 도 32는 '계층 파일의 새로운 자식 파일 만들기' 과정을 설명하기 위한 흐름도를 보여 준다.

계층 파일의 자식 파일은 부모 파일과 같은 디렉토리에 존재한다. 단계 S2301에서는 부모 파일의 경로명(PathName)과 자식 파일의 파일명(FileName)을 설정한다. 부모 파일과 자식 파일이 같은 디렉토리 내에 만들어지지만, 이것은 한정적인 것이 아니며, 두 개의 파일이 서로 다른 디렉토리에 존재할 수도 있다.

이 과정은 부모 파일의 경로명에 따라 부모 파일을 검색하고 디렉토리 엔트리를 획득하는 단계 S2302에서 시작한다. 단계 S2304에서는 부모 파일을 찾을 수 있는지를 판단한다. 찾을 수 있으면, 단계 S2304로 진행하고, 그렇지 않으면 단계 S2305로 진행하여 오류 처리를 수행한다. 단계 S2306에서는 검색된 파일이 계층 파일의 부모 파일인지를 판단한다. 할당 형태 'ParentAllocType'의 값이 'FFFFFFFFh'이면, 단계 S2307로 진행하고, 그렇지 않으면 단계 S2308로 진행하여 오류 처리를 수행한다.

단계 S2307에서는, 같은 디렉토리 내에 파일명 'FileName'으로 자식 파일을 만든다. 단계 S2309에서는 파일이 만들어졌는지를 확인한다. 파일이 만들어졌으면, 단계 S2310으로 진행하고, 그렇지 않으면 단계 S2311로 진행하여 오류 처리를 수행한다.

단계 S2310에서는 '(12) 계층 파일의 새 프레그먼트 획득' 과정을 수행하고 현재 파일의 프레그먼트 어드레스(FragmentAddress) 상에 값 '0'을 설정하고 부모 파일의 선두 종료 위치를 할당 형태(AllocType)로 설정한다. 단계 S2312에서, 새로 획득된 프레그먼트의 어드레스(FragmentAddress)를 자식 파일의 선두 종료 위치(Location)로 설정한다. 단계 S2313은 자식 파일의 엔트리를 갱신한다. 과정은 단계 S2314에서 종료된다.

앞서 설명한 실시예에서, 클러스터와 프레그먼트는 FAT 체인으로 각각 표현하였다. 그러나, 이에 한정된 것이 아니라, 비트맵을 이용하여 볼륨 할당을 위한 저장 영역의 이용에 관한 정보를 관리하고, 다른 방법으로 파일 구조에 관한 정보를 관리할 수 있다. 이 경우, 클러스터 관리와 프레그먼트 관리에 비트맵을 사용하여 클러스터 단위로 이루어지는 할당과 프레그먼트 단위로 이루어지는 할당이 모두가능하도록 할 수 있다. 이는, 앞서 설명한 실시예와 동일한 효과를 나타낸다.

클러스터 비트맵을 사용하는 경우, 각 클러스터 내에 사용중인 프레그먼트의 개수와 같은 정보를 클러스터 비트맵에 추가시킬 수 있다. 이렇게 함으로써, 상위 계층의 제 1 관리 정보로부터 하위 계층의 제 1 관리 정보를 얻을 수 있다. 이 것은 청구항 14에 기재된 특징을 구현한다.

상기 실시예에서, 파일은 클러스터와 프레그먼트의 두 가지 계층에서 관리된다. 이 것은 실시예의 설명을 위한 것일 뿐이며, 본 발명은 각 프레그먼트를 보다 작은 단위로 세분하여 대형 볼륨을 다층의 관리 정보를 이용하여 관리할 수 있도록 수행할 수도 있다.

상기 실시예에서, 정보 기록 매체의 전체 저장 영역이 하나의 볼륨으로 다루어진다. 또한, 물리적으로 별개인 정보 기록 매체를 하나의 볼륨으로 논리적으로 이용할 수 있고, 단일 정보 기록 매체의 영역을 복수개의 볼륨으로 세분할 수 있다.

상기 실시예는 모든 클러스터의 프레그먼트 FAT 엔트리들을 이용한다. 그러나, 각 클러스터에 대한 프레그먼트 FAT의 엔트리들을 보여주는 관리 테이블을 이용함으로써 이를 참조하여 프레그먼트 FAT 엔트리들을 간접적으로 얻을 수 있다. 이와 같은 경우, 프레그먼트 단위로 사용되고 있지 않은 클러스터들은 프레그먼트 FAT 엔트리를 갖지 않는다. 그러므로, 프레그먼트와 관련한 클러스터가 프레그먼트 단위로 새로 사용될 때에만 프레그먼트 FAT 엔트리들이 볼륨 내에 형성된다.

이와 같은 변형예는 프레그먼트 FAT의 크기를 감소시켜 볼륨의 사용 효율을 증가시킨다. 또한, 필수적으로 사용되는 프레그먼트 FAT들을 집중시킬 수 있다. 이로써, 이차 기록 장치(503)에 효율적으로 접근할 수 있게 된다.

상기 실시예에서, 각 파일에 대한 볼륨 할당이 바뀌지 않는다. 그러나, 각 파일에 대한 할당 단위를 바꾸어 줄 수 있다. 파일 내에서 할당 단위를 바꾸어 주기 위해서는, 클러스터 FAT의 엔트리가 엔트리 값의 위치를 나타내도록 엔트리 값의 어드레스 크기를 변화시킨다.

상기 실시예에서, 클러스터의 크기와 프레그먼트의 크기는 일정하다. 그러나, 이에 한정적인 것이 아니라, 가변 길이 클러스터 및 프레그먼트를 이용하여 파일을 구성할 수 있다. 가변 길이 클러스터를 구현하기 위하여, 클러스터 내에 사용되는 영역을 정의하는 시작 오프셋과 종료 오프셋을 나타내는 필드를 클러스터 FAT 엔트리에 첨가시킨다. 마찬가지로, 프레그먼트 내에 사용되는 영역을 정의하는 시작 오프셋과 종료 오프셋을 나타내는 프레그먼트 엔트리 필드를 첨가하여 가변 길이 프레그먼트를 얻는다. 이로써, 파일 내의 데이터를 복사할 필요 없이, 파일 병합이나 파일 내의 데이터 삭제와 같은 편집 작업을 효과적으로 수행할 수 있다.

본 발명의 계층적 관리 파일 장치(510)를 이용하는 도 1에 나타낸 전자 장치 또는 시스템 장치는 본 발명의 실시예를 설명하기 위한 것이다. 본 발명의 계층적 관리 파일 장치는 고속으로 연속 기록 또는 재생되는 파일들과 불연속적으로 처리되는 파일들을 동시에 다룰 수 있는 형태의 전자 장치나 시스템에 사용할 수 있다.

본 발명의 계층적 관리 파일 장치와, 이 파일 장치를 이용한 전자 장치 또는 시스템에 있어서, 파일은 파일에 할당된 서로 다른 계층의 다수의 영역 형태에 따라서 관리된다. 영상 및 음향의 실시간 기록 또는 재생이 필요한 경우, 기록 단위 길이가 긴 연속적인 기록 영역을 할당함으로써 손실에 의한 파일 엑세스 속도 감소를 방지할 수 있다.

이와 반대로, 보다 작은 기록 단위 길이의 영역을 크기가 작은 파일에 할당함으로써 기록 영역의 용량이 낭비되는 것을 방지한다.

또한, 단일 기록 단위 길이에 따라 할당이 이루어진 종래의 경우와 비교하여 기록 단위 길이가 큰 상위 계층의 영역에 대한 제 1 관리 정보의 데이터 크기를 감소시킬 수 있다. 이와 반대로, 제 1 관리 정보의 메모리 크기를 줄임으로써 그에 대한 이용을 용이하게 할 수 있다. 이와 동시에, 처리 동작시에 CPU에 대한 가해지는 부하와 디스크에 접근할 때 발생하는 부하를 크게 줄일 수 있다.

보다 큰 기록 단위 길이의 영역 단위로 할당이 이루어지는 경우, 새로 할당되는 사용하지 않은 기록 영역을 쉽게 검색할 수 있다. 또한, 보다 큰 기록 단위 길이 단위로 제 1 관리 정보와 제 2 관리 정보를 이용함으로써 손실을 줄이면서도 할당을 최적화할 수 있다.

할당된 기록 영역은 제 1 관리 정보를 갖는다. 상위 계층의 제 1 관리 정보는 하위 계층의 영역의 이용 상태에 관한 통계 정보를 포함할 수 있다. 파일을 하위 계층의 영역 단위로 기록하는 경우, 하위 계층의 영역 상태를 상위 계층의 제 1 관리 정보로부터 알 수 있으므로 하위 계층의 제 1정보를 이용할 필요가 없게 된다. 그러므로, 보다 작은 기록 단위 길이의 영역 단위로 쓰기 동작을 하는 경우에도 손실을 줄이면서 파일을 효율적으로 기록할 수 있다.

부모 파일과 자식 파일을 갖는 계층 파일을 이용하면 자식 파일에 관한 정보를 보다 큰 기록 단위 길이에서 할당된 영역으로 집중시킬 수 있다. 따라서, 한꺼번에 전체 계층 파일에 접근할 때 엑세스 속도가 증가된다. 또한, 계층 파일의 원리를 이용하여 부모 파일이 디렉토리로 작용하는 속성을 갖는다. 디렉토리로서 부모 파일을 이용함으로써, 보다 큰 기록 단위 길이로 할당된 영역에 디렉토리 내의 모든 파일들을 집중시킬 수 있다. 이렇게 하여, 디렉토리 내의 파일들에 대한 엑세스 속도를 높일 수 있다. 또한, 디렉토리와 함께 파일들을 삭제하는 것이 바람직한 경우, 보다 큰 기록 단위 길이의 영역을 남겨두는 방식으로 삭제하면 사용되지 않은 기록 영역들이 분산되는 것을 막을 수 있다.

계층 파일의 원리를 이용하여 보다 작은 기록 단위 길이의 영역들을 할당하여 보다 큰 기록 단위 길이에서 할당된 영역을 구성하도록 할 수 있다. 그러므로, 보다 작은 기록 단위 길이의 복수개의 연속적인 영역의 배치를 새로운 할당 단위로 사용할 수 있다.

보다 큰 기록 단위 길이의 영역에 대한 제 1 관리 정보의 데이터 크기는 작게 할 수 있다. 따라서, 파일을 재생하고 기록하기 전에 파일의 읽기 및 쓰기 속도를 물리적인 파일 배열에 따라 쉽게 계산할 수 있다. 그러므로, 엑세스 속도가 높은 계층적 관리 파일 장치를 얻을 수 있다.

발명의 요약

본 발명에 따라 정보 기록 매체가 복수개의 기록 단위 길이를 갖는 영역으로 나누어지고, 각 계층 수준의 영역이 낮은 계층의 복수 영역으로 이루어지는 계층적 관리 파일 장치를 제공할 수 있다.

(5) 청구의 범위

청구항 1. 정보 기록 매체의 기록 영역이 복수개의 기록 단위 길이를 갖는 영역으로 계층적으로 분할되고, 각 계층의 영역이 하위 계층의 복수개의 영역으로 이루어지고, 파일이 제 1 관리 정보와 제 2 관리 정보의 관리하에 계층들의 영역 단위로 기록 매체의 파일 기록 영역에 분할 방식으로 기록되고, 파일이 제 1 관리 정보와 제 2 관리 정보에 따라 계층들의 영역을 계층적으로 관리함으로써 기록 매체로부터 재생되는 계층적 관리 파일 장치에 있어서,

제 1 관리 정보와 제 2 관리 정보를 기록하고, 제 1 관리 정보와 제 2 관리 정보의 재생을 가능케 하고; 상기 제 1 관리 정보가 각 계층에 대하여 각 계층의 영역의 연결 상태를 나타내는 연결 정보와 영역의 사용 상태를 나타내는 사용 정보를 포함하고;

상기 제 2 관리 정보가 상기 파일의 속성, 파일의 기록 및 재생에 사용되는 계층의 등급, 파일의 형태, 그리고 파일의 기록 및 재생이 시작되는 시작 계층의 영역 위치를 포함하는 정보 기록 매체와;

각 계층의 제 1 관리 정보와 제 2 관리 정보를 참조하여 파일의 기록 및 재생을 관리하고, 상기 파일을 각 계층들의 영역에 기록하기 위하여 하위 계층의 제 1 관리 정보를 참조하여 하위 계층의 영역들을 파일에 할당하고; 하위 계층에 빈 영역이 존재하지 않으면 상위 계층의 제 1 관리 정보를 이용한 후에 상위 계층에 속하는 하위 계층의 제 1 관리 정보를 이용하여 상기 영역들에 파일을 나누도록 하고;

파일의 특성에 따라 계층을 선택적으로 이용하는 소프트웨어 제어 수단을 포함하는 계층적 관리 파일 장치.

청구항 2. 제 1 항에 있어서, 소프트웨어 제어 수단이 각 계층에 주어진 번호가 증가하는 순서대로 제 1 관리 정보, 제 2 관리 정보 및 상기 파일 기록 영역을 각 계층의 영역들에 기록 및 재생하도록 하고; 상기 영역들에는 오름차순의 번호가 부여되는 계층적 관리 파일 장치.

청구항 3. 제 1 항에 있어서, 소프트웨어 제어 수단이 제 2 관리 정보를 같은 계층의 영역들에 기록하도록 제어하고;

제 2 관리 정보가 적어도 상기 파일의 속성들을 나타내는 제 1 종의 정보와, 파일의 기록 및 재생에 이용되는 계층과 파일의 형태를 나타내는 제 2 종의 정보와, 파일의 기록 및 재생이 시작되는 계층의 영역들의 위치를 나타내는 제 3 종의 정보를 포함하고;

제 2 종의 정보에 따라 상기 파일이 계층 파일이 아니라고 판단되면, 제 3 종의 정보와 결정된 계층을 이용하여 상기 파일의 기록 및 재생을 수행하는 계층적 관리 파일 장치.

청구항 4. 제 1 항에 있어서, 소프트웨어 제어 수단이 제 2 관리 정보를 같은 계층의 영역들에 기록하도록 제어하고;

제 2 관리 정보가 적어도 상기 파일의 속성들을 나타내는 제 1 종의 정보와, 파일의 기록 및 재생에 이용되는 계층과 파일의 형태를 나타내는 제 2 종의 정보와, 파일의 기록 및 재생이 시작되는 계층의 영역들의 위치를 나타내는 제 3 종의 정보를 포함하고;

제 2 종의 정보에 따라 상기 파일이 부모 파일과 자식 파일로 구성되는 계층 파일이라고 판단되면, 부모 파일의 계층의 시작 영역의 위치를 나타내는 부모 파일의 제 3 종의 정보와 일치하는 자식 파일의 제 2 종의 정보를 검색하여 자식 파일을 결정하고, 자식 파일의 제 3 종의 정보에 따라 상기 자식 파일의 기록 및 재생이 시작되는 계층의 영역의 위치를 결정하도록 소프트웨어 제어 수단이 제어한 다음에;

결정된 자식 파일의 제 1 관리 정보를 이용하여, 자식 파일이 사용한 하위 계층의 영역들이 부모 파일이 사용한 상위 계층의 영역에 반드시 포함되도록 소프트웨어 제어 수단이 제어하는 계층적 관리 파일 장치.

청구항 5. 제 1 항에 있어서, 제 1 관리 정보가 각 계층의 영역들의 연결 상태를 나타내는 연결 정보와 각 계층의 영역들의 사용 상태를 나타내는 사용 정보를 갖는 제 4 종의 정보와, 그리고 상기 계층보다 낮은 계층의 영역들의 사용 상태를 나타내는 제 5 종의 정보를 포함하고;

제 4 종의 정보에 따라 상기 계층의 영역 다음에 기록 및 재생에 사용될 영역을 결정하여 상기 파일에 할

당된 영역들을 기록 및 재생에 연속적으로 사용할 수 있도록 소프트웨어 제어 수단이 제어하고;

제 5 종의 정보에 따라 하위 계층의 영역을 파일에 할당할지를 결정하도록 소프트웨어 제어 수단이 제어하는 계층적 관리 파일 장치.

청구항 6. 제 1 항에 있어서, 하위 계층의 제 1 관리 정보가 하위 계층의 영역들의 연결 상태를 나타내는 연결 정보와, 하위 계층의 영역들의 사용 상태를 나타내는 사용 정보를 포함하고;

하위 계층의 상기 영역들이 속하는 상위 계층의 영역들의 연결 상태를 나타내는 연결 정보와, 상위 계층의 영역들의 사용 상태를 나타내는 사용 정보를 포함하는 계층적 관리 파일 장치.

청구항 7. 제 1 항에 있어서, 각 계층의 제 1 관리 정보가 상기 파일의 종료점을 나타내는 정보를 포함하는 계층적 관리 파일 장치.

청구항 8. 제 1 항에 있어서, 정보 기록 매체가 초기화되면 가장 큰 기록 단위 길이를 갖는 최상위 계층의 영역들만을 할당하도록 소프트웨어 제어 수단이 제어 동작을 하는 계층적 관리 파일 장치.

청구항 9. 제 1 항에 있어서, 소프트웨어 제어 수단이 상기 파일의 연속적인 정보 단위를 하위 계층의 영역들에 부여하도록 동작 가능한 계층적 관리 파일 장치.

청구항 10. 제 1 항에 있어서, 임의 계층의 새로운 영역이 임의 파일에 대하여 보존되는 경우와 상기 파일이 새로운 영역에 기록되는 경우, 소프트웨어 제어 수단이 상기 계층의 영역들에 대한 연결 정보 및 사용 정보를 갖는 제 4 종의 정보와 하위 계층의 영역들의 사용 상태를 나타내는 제 5 종의 정보를 각 계층에 대하여 갱신하는 계층적 관리 파일 장치.

청구항 11. 제 1 항에 있어서, 임의 계층의 새로운 영역이 임의 파일에 대하여 보존되는 경우와 상기 파일이 새로운 영역에 기록되는 경우, 소프트웨어 제어 수단이 상기 파일의 속성을 나타내는 제 1 종의 정보를 갱신하고 제 2 관리 정보에 포함된 제 2 종의 정보와 제 3 종의 정보를 갱신하고;

제 1 종의 정보가 상기 파일의 기록 및 재생에 사용되는 계층을 나타내고;

제 2 종의 정보가 파일의 기록 및 재생이 시작되는 계층의 영역의 위치를 나타내는 계층적 관리 파일 장치.

청구항 12. 제 1 항에 있어서, 소프트웨어 제어 수단이 파일을 기록하기 위한 새로운 영역을 할당하여 제 1 관리 정보와 제 2 관리 정보를 갱신할 때에는 제 1 관리 정보와 제 2 관리 정보를 참조하고, 파일이 가장 최근에 기록된 현재 계층의 영역의 위치를 이용하여 제 1 관리 정보와 제 2 관리 정보를 갱신할 때에는 현재 계층의 영역이 속하는 최상위 계층의 영역의 제 1 관리 정보를 참조하여 전체 계층 중에서 최상위 계층의 사용하지 않은 영역을 검색하고;

소프트웨어 수단이 연속적인 하위 계층에 대하여 최상위 계층의 영역의 검색과 유사한 검색을 수행하고, 파일에 첨가될 정보를 현재의 계층의 영역에 가장 가까운 검색 영역으로 할당하는 계층적 관리 파일 장치.

청구항 13. 제 1 항에 있어서, 소프트웨어 제어 수단이 계층 파일을 기록하기 위한 새로운 영역을 할당하여 제 1 관리 정보와 제 2 관리 정보를 갱신할 때에는 제 1 관리 정보와 제 2 관리 정보를 참조하고, 계층 파일이 가장 최근에 기록된 현재 계층의 영역의 위치를 이용하여 제 1 관리 정보와 제 2 관리 정보를 갱신할 때에는 현재 계층의 영역이 속하고 부모 파일에 속하는 최상위 계층의 영역의 제 1 관리 정보를 참조하여 전체 계층 중에서 부모 파일에 속하는 최상위 계층의 사용하지 않은 영역을 검색하고;

소프트웨어 수단이 연속적인 하위 계층에 대하여 부모 파일에 속하는 최상위 계층의 사용하지 않은 영역의 검색과 유사한 검색을 수행하고, 현재 계층의 영역에 가장 가까운 검색 영역에 상기 계층의 자식 파일의 정보를 기록하는 계층적 관리 파일 장치.

청구항 14. 제 1 항에 있어서, 각 계층에 대하여 제 1 관리 정보가 각 계층의 영역들의 사용 상태를 나타내는 사용 정보와, 상기 계층보다 낮은 계층의 영역들의 사용 상태를 나타내는 사용 정보를 포함하고;

각 계층의 영역들의 연결 상태를 나타내는 연결 정보가 제 2 관리 정보나 제 1 관리 정보와 제 2 관리 정보가 기록되어 있는 영역과 다른 기록 영역에 주어진 제 3 관리 정보에 포함되는 계층적 관리 파일 장치.

청구항 15. 정보 기록 매체의 기록 영역이 복수개의 기록 단위 길이를 갖는 영역으로 계층적으로 분할되고, 각 계층의 영역이 하위 계층의 복수개의 영역으로 이루어지고, 파일이 제 1 관리 정보와 제 2 관리 정보의 관리하에 계층들의 영역 단위로 기록 매체의 파일 기록 영역에 분할 방식으로 기록되고, 파일이 제 1 관리 정보와 제 2 관리 정보에 따라 계층들의 영역을 계층적으로 관리함으로써 기록매체로부터 재생되는 형태의 전자 장치에 있어서,

제 1 관리 정보와 제 2 관리 정보를 기록하고, 제 1 관리 정보와 제 2 관리 정보의 재생을 가능케 하고;

상기 제 1 관리 정보가 각 계층에 대하여 각 계층의 영역의 연결 상태를 나타내는 연결 정보와 영역의 사용 상태를 나타내는 사용 정보를 포함하고;

상기 제 2 관리 정보가 상기 파일의 속성, 파일의 기록 및 재생에 사용되는 계층의 등급, 파일의 형태, 그리고 파일의 기록 및 재생이 시작되는 시작 계층의 영역 위치를 포함하는 정보 기록 매체와;

각 계층의 제 1 관리 정보와 제 2 관리 정보를 참조하여 파일의 기록 및 재생을 관리하며, 상기 파일을 각 계층들의 영역에 기록하기 위하여 하위 계층의 제 1 관리 정보를 참조하여 하위 계층의 영역들을 파일에 할당하고, 하위 계층에 빈 영역이 존재하지 않으면 상위 계층의 제 1 관리 정보를 이용한 후에 상위 계층에 속하는 하위 계층의 제 1 관리 정보를 이용하여 상기 영역들에 파일을 나누도록 하고, 파일의 특성에 따라 계층을 선택적으로 이용하는 소프트웨어 제어 수단으로 이루어진 계층적 관리 파일 장치를 포함하는 전자 장치.

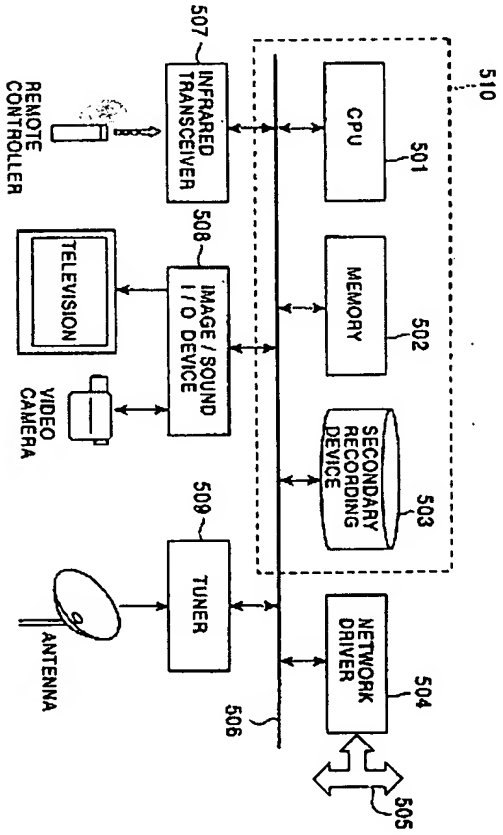
청구항 16. 계층적으로 관리된 파일을 정보 기록 매체의 기록 영역에 기록하고 이로부터 재생하는 방법에 있어서,

상기 기록 영역이 복수개의 기록 단위 길이를 갖는 영역들로 계층적으로 분할되고, 각 계층의 영역이 하위 계층의 복수개의 영역들로 이루어지고, 파일이 제 1 관리 정보와 제 2 관리 정보의 관리하에 계층의 영역 단위로 기록 매체의 파일 기록 영역에 분할 방식으로 기록되고, 파일이 제 1 관리 정보와 제 2 관리 정보에 따라 계층의 영역들을 계층적으로 관리함으로써 기록 매체로부터 재생되고, 정보 기록 매체가 제 1 관리 정보와 제 2 관리 정보를 기록하고, 각 계층에 대하여, 제 1 관리 정보가 각 계층의 영역들의 연결 상태를 나타내는 연결 정보와 영역들의 사용 상태를 나타내는 사용 정보를 포함하고, 제 2 관리 정보가 상기 파일의 속성들과, 파일의 기록 및 재생에 사용되는 계층의 등급과, 파일의 기록 및 재생이 시작되는 시작 계층의 영역의 위치를 포함하고, 각 계층의 제 1 관리 정보와 제 2 관리 정보를 이용하여, 상기 파일을 계층의 영역들에 기록하기 위하여 하위 계층의 제 1 관리 정보를 참조하여 상기 파일의 하위 계층의 영역들을 할당하는 단계와;

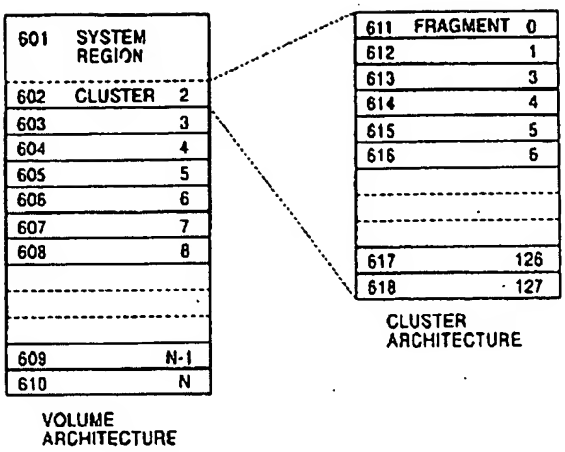
하위 계층의 빈 영역이 존재하지 않으면, 상위 계층의 제 1 관리 정보를 참조한 후에, 상위 계층에 속하는 하위 계층의 제 1 관리 정보를 통하여 다시 검색을 수행하여 상기 영역들에 파일을 할당하고; 파일의 특성에 따라 계층을 선택적으로 이용하면서 정보 기록 매체로 파일을 기록하고 정보 기록 매체로부터 파일을 재생하는 단계를 포함하는 계층적으로 관리된 파일을 기록하고 재생하는 방법.

도면

도면1



도면2



도면3

1. UNUSED CLUSTER

32 BITS	
701	0 (IDENTIFIER)

2. CLUSTER IN USE ON FRAGMENT BASIS

25 BITS		7 BITS	
702	0 (IDENTIFIER)	703	NUMBER OF FRAGMENTS IN USE (1 TO 127)

3. CLUSTER IN USE ON FRAGMENT BASIS (ALL FRAGMENTS USED)

25 BITS		7 BITS	
704	1 (IDENTIFIER)	705	0 (IDENTIFIER)

4. CLUSTER ALLOCATED TO FILE

25 BITS		7 BITS	
706	CLUSTER NUMBER (2 OR GREATER)	707	0 (IDENTIFIER)

5. CLUSTER ALLOCATED TO FILE (USED ON FRAGMENT BASIS IN HIERARCHICAL FILE)

25 BITS		7 BITS	
708	CLUSTER NUMBER (2 OR GREATER)	709	NUMBER OF FRAGMENTS IN USE (0 TO 127)

도면4

1. UNUSED FRAGMENT

32 BITS
801 0 (IDENTIFIER)

2. FRAGMENT IN USE ON CLUSTER BASIS

25 BITS	7 BITS
802 1 (IDENTIFIER)	803 0 (IDENTIFIER)

3. FRAGMENT ALLOCATED TO FILE

25 BITS	7 BITS
804 CLUSTER NUMBER (2 OR GREATER)	805 FRAGMENT NUMBER (1 TO 127)

도면5a

901 FILE1.DAT	902 00000000h	903 0000002h,00h	904	1032091
905 FILE2.DAT	906 00000000h	907 0000007h,00h	908	65536
909 FILE3.DAT	910 00000000h	911 0000017h,00h	912	65537
913 FILE4.DAT	914 00000001h	915 0000012h,00h	916	1518
917 FILE5.DAT	918 00000001h	919 0000012h,04h	920	1024
921 FILE6.DAT	922 FFFFFFFFh	923 0000018h,00h	924	196608
925 FILE7.DAT	926 00000018h	927 0000018h,00h	928	
929 FILE8.DAT	930 00000018h	931 0000018h,01h	932	

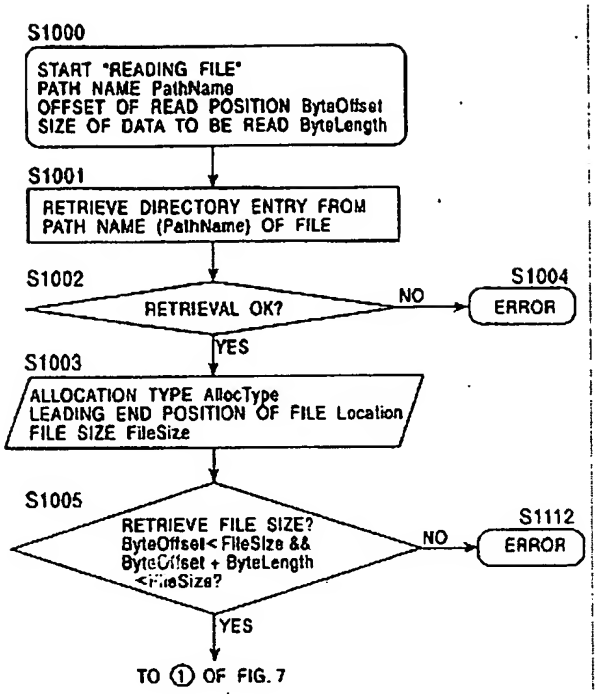
도면5b

933	934	935 0000004h,00h	936 00000000h,00h
937 0000005h,00h	938 0000008h,00h	939 1FFFFFFh,00h	940 1FFFFFFh,00h
941 0000009h,00h	942 000000Ah,00h	943 00000008,00h	944 000000Ch,00h
945 000000Dh,00h	946 000000Eh,00h	947 000000Fh,00h	948 0000010h,00h
949 0000011h,00h	950 0000014h,00h	951 00000000h,05h	952 00000000h,01h
953 0000015h,00h	954 0000016h,00h	955 1FFFFFFh,00h	956 0000006h,00h
957 0000018h,03h	958 000002Ah,03h	959 1FFFFFFh,02h	960 00000000h,00h
961 00000000h,00h	962 00000000h,00h	963 00000000h,00h	964 00000000h,00h

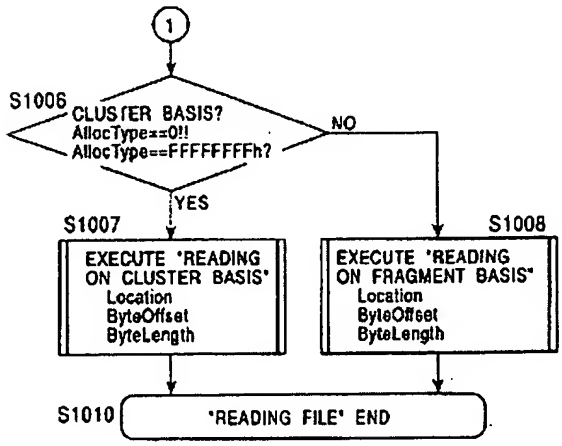
도면5c

965 0000001h,00h	966 0000001h,00h	967 0000001h,00h	968 0000001h,00h
969 0000012h,01h	970 0000012h,02h	971 0000012h,FFh	972 0000013h,01h
973 00000000h,00h	974 00000000h,00h	975 00000000h,00h	976 1FFFFFFh,00h
977 00000000h,00h	978 1FFFFFFh,00h	979 00000000h,00h	980 00000000h,00h
981 0000018h,02h	982 0000019h,00h	983 000001Ah,02h	984 00000000h,00h
985 0000019h,01h	986 0000019h,02h	987 000001Ah,01h	988 00000000h,00h
989 00000000h,00h	990 1FFFFFFh,00h	991 1FFFFFFh,00h	992 00000000h,00h
993 0000001h,00h	994 0000001h,00h	995 0000001h,00h	996 0000001h,00h

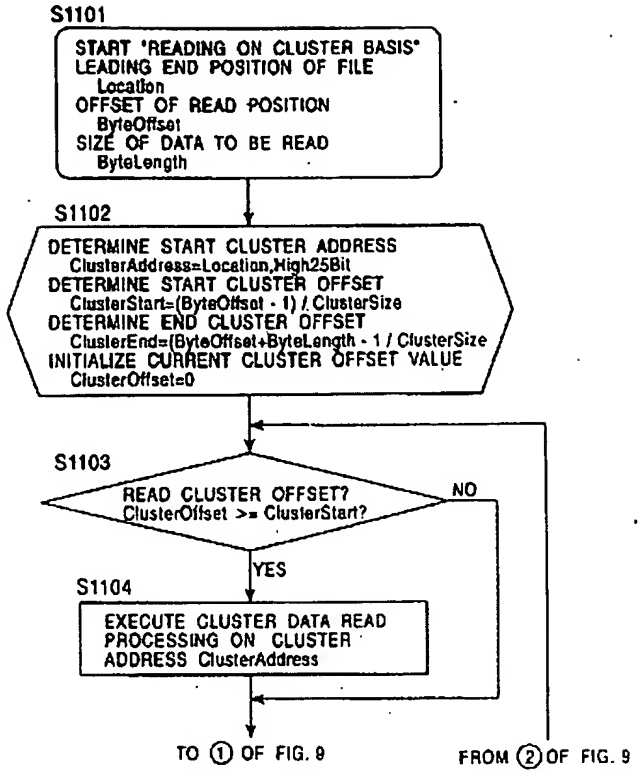
도면6


S1000
START "READING FILE"
PATH NAME PathName
OFFSET OF READ POSITION ByteOffset
SIZE OF DATA TO BE READ ByteLength
S1001
RETRIEVE DIRECTORY ENTRY FROM PATH NAME (PathName) OF FILE
S1002
RETRIEVAL OK?
NO
S1004
ERROR
YES
S1003
ALLOCATION TYPE AllocType
LEADING END POSITION OF FILE Location
FILE SIZE FileSize
S1005
RETRIEVE FILE SIZE?
ByteOffset< FileSize &&
ByteOffset + ByteLength
<FileSize?
NO
S1112
ERROR
YES
TO ① OF FIG. 7

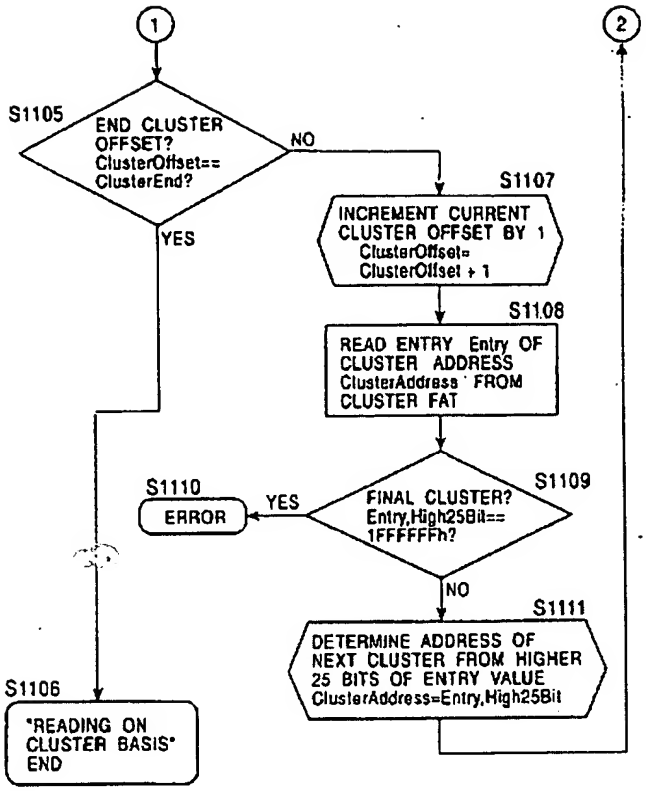
도면7


①
S1006
CLUSTER BASIS?
AllocType==0!!
AllocType==FFFFFFFFh?
NO
YES
S1007
EXECUTE "READING ON CLUSTER BASIS"
Location
ByteOffset
ByteLength
S1008
EXECUTE "READING ON FRAGMENT BASIS"
Location
ByteOffset
ByteLength
S1010
"READING FILE" END

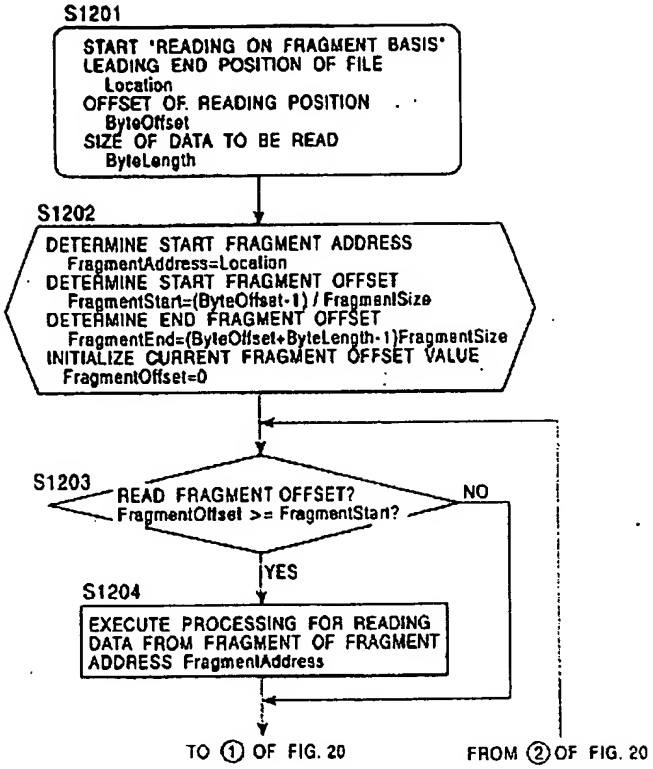
도면8


S1101
START 'READING ON CLUSTER BASIS'
LEADING END POSITION OF FILE
Location
OFFSET OF READ POSITION
ByteOffset
SIZE OF DATA TO BE READ
ByteLength
S1102
DETERMINE START CLUSTER ADDRESS
ClusterAddress=Location,High25Bit
DETERMINE START CLUSTER OFFSET
ClusterStart=(ByteOffset - 1) / ClusterSize
DETERMINE END CLUSTER OFFSET
ClusterEnd=(ByteOffset+ByteLength - 1 / ClusterSize
INITIALIZE CURRENT CLUSTER OFFSET VALUE
ClusterOffset=0
S1103
READ CLUSTER OFFSET?
ClusterOffset >= ClusterStart?
NO
YES
S1104
EXECUTE CLUSTER DATA READ
PROCESSING ON CLUSTER
ADDRESS ClusterAddress
TO ① OF FIG. 9
FROM ② OF FIG. 9

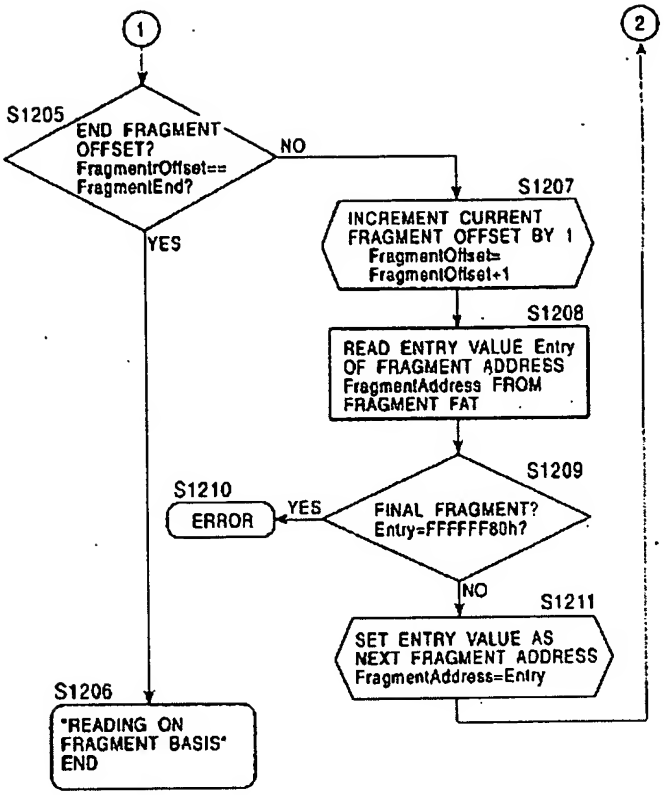
도면9


1
2
S1105
END CLUSTER OFFSET?
ClusterOffset==
ClusterEnd?
NO
YES
S1107
INCREMENT CURRENT CLUSTER OFFSET BY 1
ClusterOffset=
ClusterOffset + 1
S1108
READ ENTRY Entry OF CLUSTER ADDRESS ClusterAddress FROM CLUSTER FAT
S1109
FINAL CLUSTER?
Entry,High25Bit==
1FFFFFFh?
YES
S1110
ERROR
NO
S1111
DETERMINE ADDRESS OF NEXT CLUSTER FROM HIGHER 25 BITS OF ENTRY VALUE
ClusterAddress=Entry,High25Bit
S1106
"READING ON CLUSTER BASIS" END

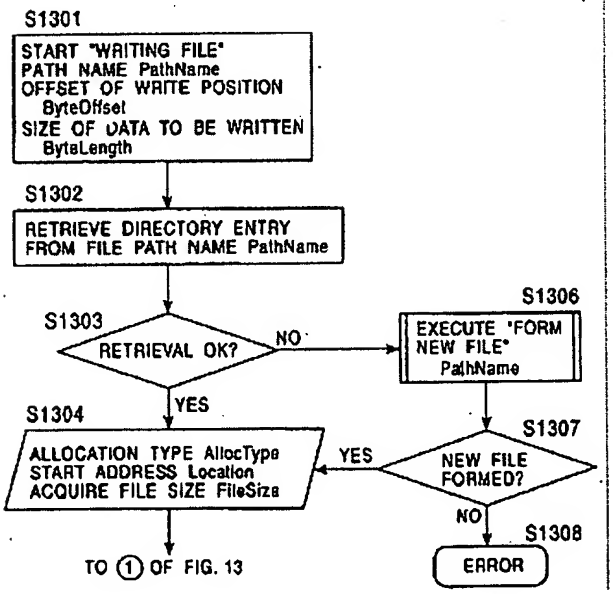
도면10


S1201
START 'READING ON FRAGMENT BASIS'
LEADING END POSITION OF FILE
Location
OFFSET OF READING POSITION
ByteOffset
SIZE OF DATA TO BE READ
ByteLength
S1202
DETERMINE START FRAGMENT ADDRESS
FragmentAddress=Location
DETERMINE START FRAGMENT OFFSET
FragmentStart=(ByteOffset-1) / FragmentSize
DETERMINE END FRAGMENT OFFSET
FragmentEnd=(ByteOffset+ByteLength-1)FragmentSize
INITIALIZE CURRENT FRAGMENT OFFSET VALUE
FragmentOffset=0
S1203
READ FRAGMENT OFFSET?
FragmentOffset >= FragmentStart?
NO
YES
S1204
EXECUTE PROCESSING FOR READING DATA FROM FRAGMENT OF FRAGMENT ADDRESS FragmentAddress
TO ① OF FIG. 20
FROM ② OF FIG. 20

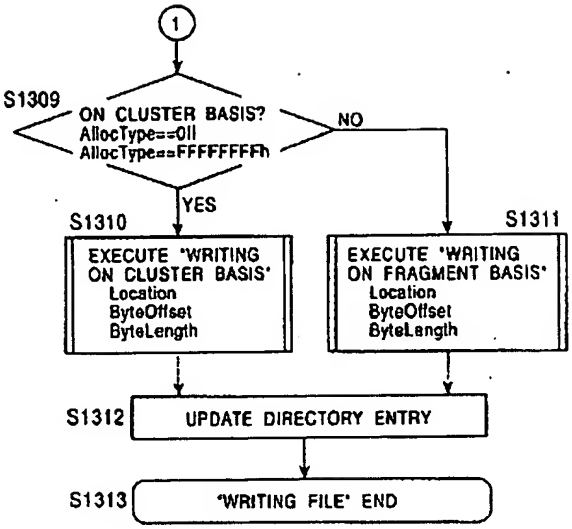
도면11


1
2
S1205
END FRAGMENT OFFSET?
FragmentrOffset==
FragmentEnd?
NO
YES
S1207
INCREMENT CURRENT FRAGMENT OFFSET BY 1
FragmentOffset=
FragmentOffset+1
S1208
READ ENTRY VALUE Entry OF FRAGMENT ADDRESS FragmentAddress FROM FRAGMENT FAT
S1209
FINAL FRAGMENT?
Entry=FFFFFF80h?
S1210
ERROR
YES
NO
S1211
SET ENTRY VALUE AS NEXT FRAGMENT ADDRESS
FragmentAddress=Entry
S1206
"READING ON FRAGMENT BASIS" END

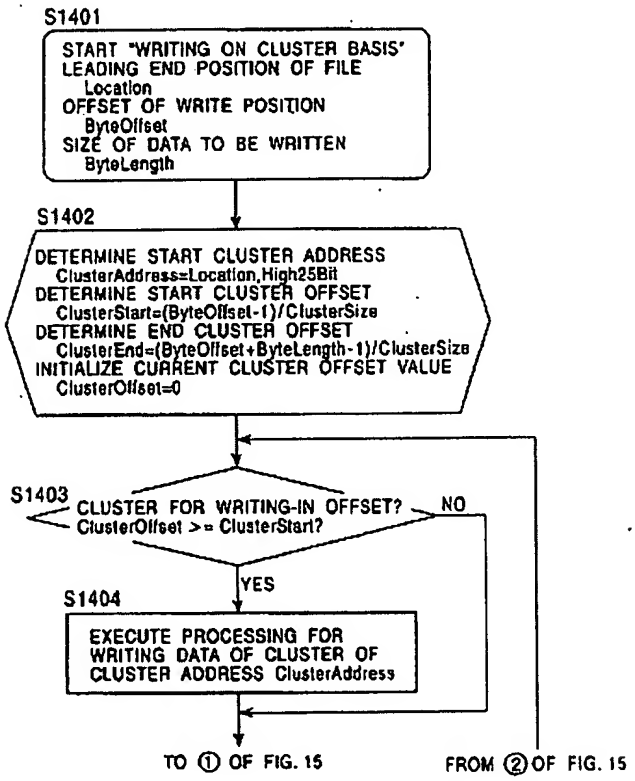
도면12


S1301
START "WRITING FILE"
PATH NAME PathName
OFFSET OF WRITE POSITION
ByteOffset
SIZE OF DATA TO BE WRITTEN
BytsLength
S1302
RETRIEVE DIRECTORY ENTRY FROM FILE PATH NAME PathName
S1303
RETRIEVAL OK?
NO
YES
S1306
EXECUTE "FORM NEW FILE"
PathName
S1304
ALLOCATION TYPE AllocType
START ADDRESS Location
ACQUIRE FILE SIZE FileSize
YES
S1307
NEW FILE FORMED?
NO
S1308
ERROR
TO (1) OF FIG. 13

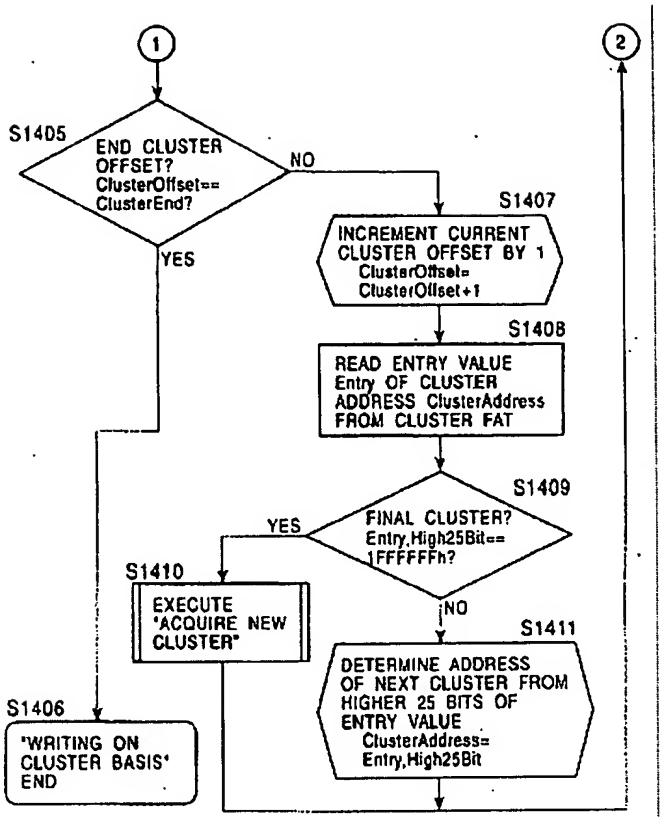
도면13


1
S1309
ON CLUSTER BASIS?
AllocType==0II
AllocType==FFFFFFFFh
NO
YES
S1310
EXECUTE "WRITING ON CLUSTER BASIS"
Location
ByteOffset
ByteLength
S1311
EXECUTE "WRITING ON FRAGMENT BASIS"
Location
ByteOffset
ByteLength
S1312
UPDATE DIRECTORY ENTRY
S1313
"WRITING FILE" END

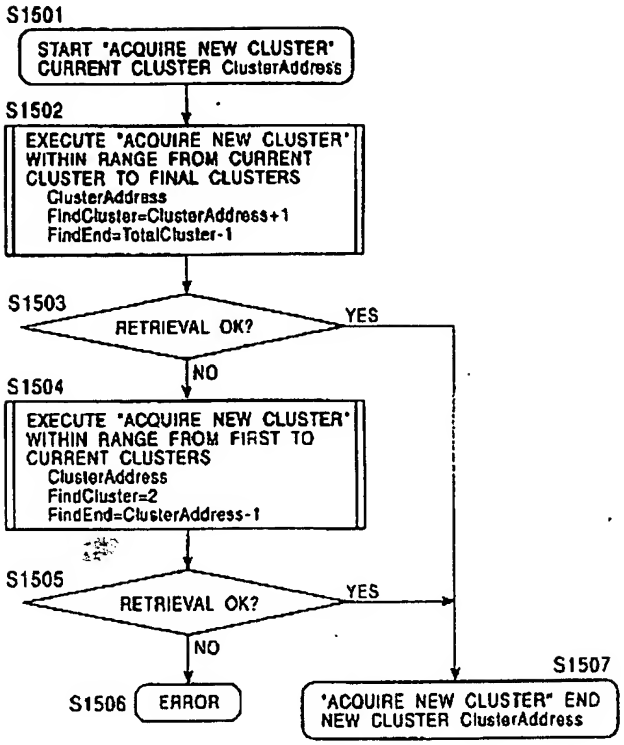
도면14


S1401
START "WRITING ON CLUSTER BASIS"
LEADING END POSITION OF FILE
Location
OFFSET OF WRITE POSITION
ByteOffset
SIZE OF DATA TO BE WRITTEN
ByteLength
S1402
DETERMINE START CLUSTER ADDRESS
ClusterAddress=Location,High25Bit
DETERMINE START CLUSTER OFFSET
ClusterStart=(ByteOffset-1)/ClusterSize
DETERMINE END CLUSTER OFFSET
ClusterEnd=(ByteOffset+ByteLength-1)/ClusterSize
INITIALIZE CURRENT CLUSTER OFFSET VALUE
ClusterOffset=0
S1403
CLUSTER FOR WRITING-IN OFFSET?
ClusterOffset >= ClusterStart?
NO
YES
S1404
EXECUTE PROCESSING FOR WRITING DATA OF CLUSTER OF CLUSTER ADDRESS ClusterAddress
TO ① OF FIG. 15
FROM ② OF FIG. 15

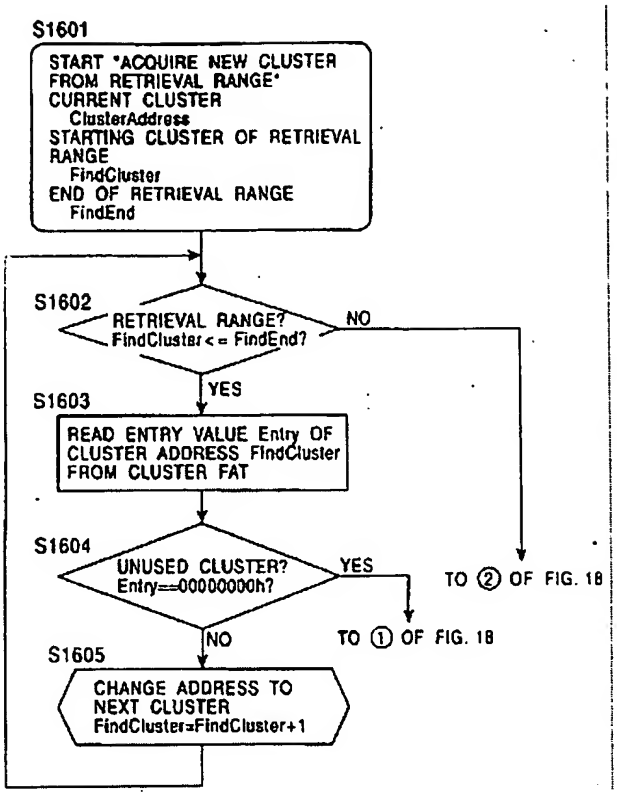
도면15


1
2
S1405
END CLUSTER OFFSET?
ClusterOffset==ClusterEnd?
NO
YES
S1407
INCREMENT CURRENT CLUSTER OFFSET BY 1
ClusterOffset=ClusterOffset+1
S1408
READ ENTRY VALUE Entry OF CLUSTER ADDRESS ClusterAddress FROM CLUSTER FAT
S1409
FINAL CLUSTER?
Entry,High25Bit==1FFFFFFh?
YES
NO
S1410
EXECUTE "ACQUIRE NEW CLUSTER"
S1411
DETERMINE ADDRESS OF NEXT CLUSTER FROM HIGHER 25 BITS OF ENTRY VALUE
ClusterAddress=Entry,High25Bit
S1406
"WRITING ON CLUSTER BASIS" END

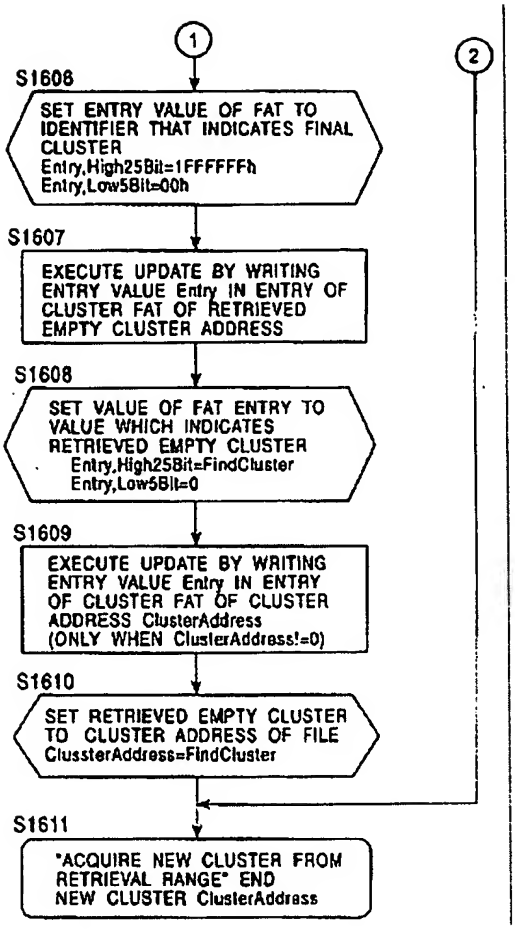
도면16


S1501
START "ACQUIRE NEW CLUSTER"
CURRENT CLUSTER ClusterAddress
S1502
EXECUTE "ACQUIRE NEW CLUSTER"
WITHIN RANGE FROM CURRENT
CLUSTER TO FINAL CLUSTERS
ClusterAddress
FindCluster=ClusterAddress+1
FindEnd=TotalCluster-1
S1503
RETRIEVAL OK?
YES
NO
S1504
EXECUTE "ACQUIRE NEW CLUSTER"
WITHIN RANGE FROM FIRST TO
CURRENT CLUSTERS
ClusterAddress
FindCluster=2
FindEnd=ClusterAddress-1
S1505
RETRIEVAL OK?
YES
NO
S1506
ERROR
S1507
"ACQUIRE NEW CLUSTER" END
NEW CLUSTER ClusterAddress

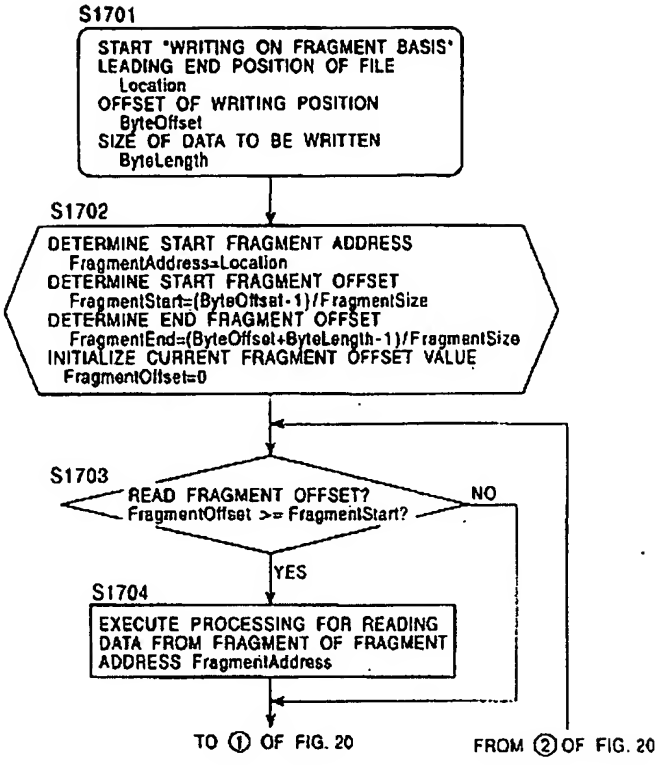
도면17


S1601
START "ACQUIRE NEW CLUSTER FROM RETRIEVAL RANGE"
CURRENT CLUSTER
ClusterAddress
STARTING CLUSTER OF RETRIEVAL RANGE
FindCluster
END OF RETRIEVAL RANGE
FindEnd
S1602
RETRIEVAL RANGE?
FindCluster <= FindEnd?
NO
YES
S1603
READ ENTRY VALUE Entry OF CLUSTER ADDRESS FindCluster FROM CLUSTER FAT
S1604
UNUSED CLUSTER?
Entry==00000000h?
YES
TO ② OF FIG. 18
NO
TO ① OF FIG. 18
S1605
CHANGE ADDRESS TO NEXT CLUSTER
FindCluster=FindCluster+1

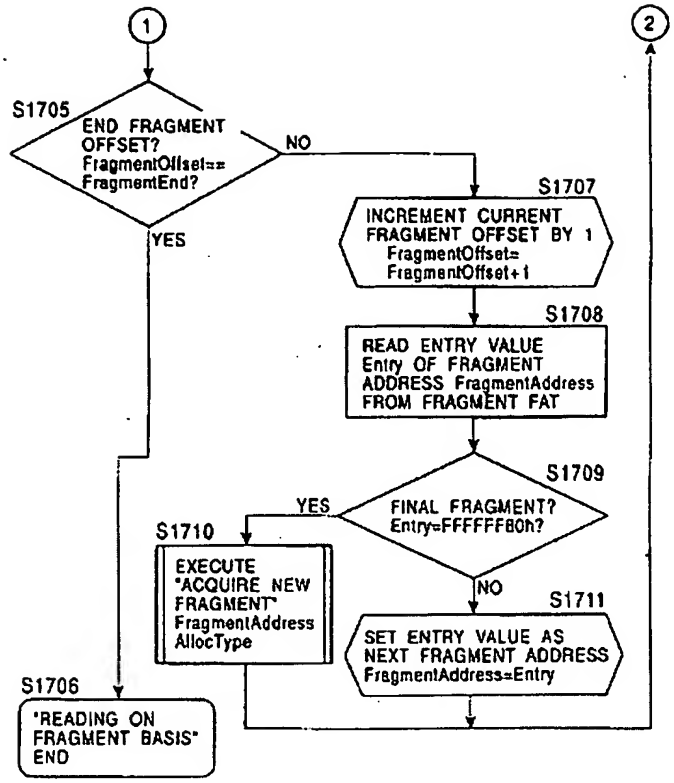
도면16


1
2
S1606
SET ENTRY VALUE OF FAT TO IDENTIFIER THAT INDICATES FINAL CLUSTER
Entry,High25Bit=1FFFFFFh
Entry,Low5Bit=00h
S1607
EXECUTE UPDATE BY WRITING ENTRY VALUE Entry IN ENTRY OF CLUSTER FAT OF RETRIEVED EMPTY CLUSTER ADDRESS
S1608
SET VALUE OF FAT ENTRY TO VALUE WHICH INDICATES RETRIEVED EMPTY CLUSTER
Entry,High25Bit=FindCluster
Entry,Low5Bit=0
S1609
EXECUTE UPDATE BY WRITING ENTRY VALUE Entry IN ENTRY OF CLUSTER FAT OF CLUSTER ADDRESS ClusterAddress
(ONLY WHEN ClusterAddress!=0)
S1610
SET RETRIEVED EMPTY CLUSTER TO CLUSTER ADDRESS OF FILE
ClussterAddress=FindCluster
S1611
"ACQUIRE NEW CLUSTER FROM RETRIEVAL RANGE" END
NEW CLUSTER ClusterAddress

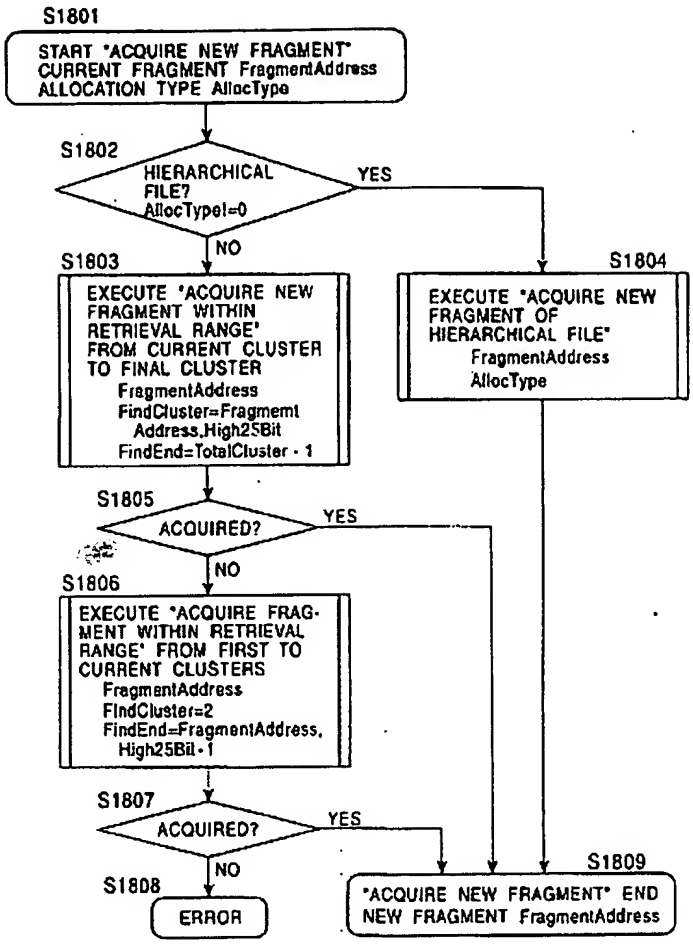
도면19


S1701
START "WRITING ON FRAGMENT BASIS"
LEADING END POSITION OF FILE
Location
OFFSET OF WRITING POSITION
ByteOffset
SIZE OF DATA TO BE WRITTEN
ByteLength
S1702
DETERMINE START FRAGMENT ADDRESS
FragmentAddress=Location
DETERMINE START FRAGMENT OFFSET
FragmentStart=(ByteOffset-1)/FragmentSize
DETERMINE END FRAGMENT OFFSET
FragmentEnd=(ByteOffset+ByteLength-1)/FragmentSize
INITIALIZE CURRENT FRAGMENT OFFSET VALUE
FragmentOffset=0
S1703
READ FRAGMENT OFFSET?
FragmentOffset >= FragmentStart?
NO
YES
S1704
EXECUTE PROCESSING FOR READING DATA FROM FRAGMENT OF FRAGMENT ADDRESS FragmentAddress
TO ① OF FIG. 20
FROM ② OF FIG. 20

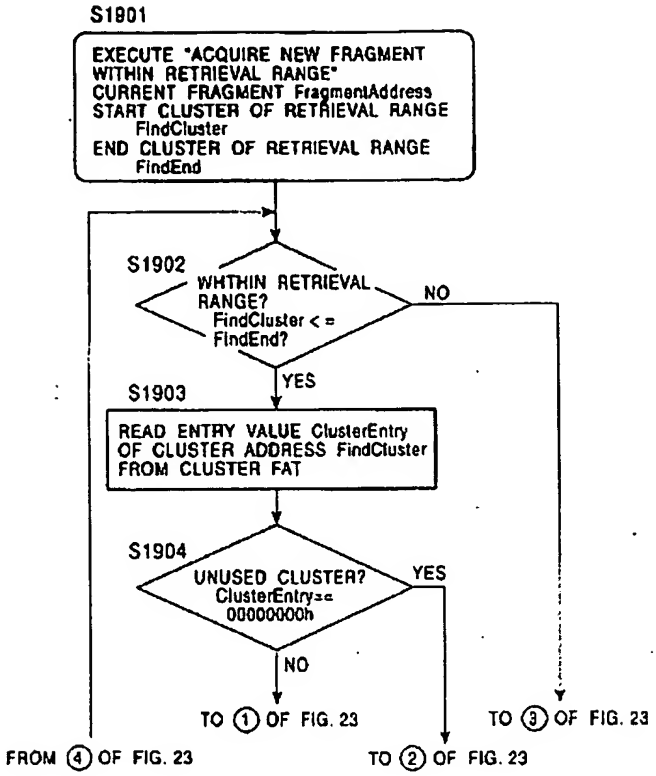
도면2


1
2
S1705
END FRAGMENT OFFSET?
FragmentOffset==FragmentEnd?
NO
YES
S1707
INCREMENT CURRENT FRAGMENT OFFSET BY 1
FragmentOffset=FragmentOffset+1
S1708
READ ENTRY VALUE Entry OF FRAGMENT ADDRESS FragmentAddress FROM FRAGMENT FAT
S1709
FINAL FRAGMENT?
Entry=FFFFFF80h?
YES
NO
S1710
EXECUTE "ACQUIRE NEW FRAGMENT" FragmentAddress AllocType
S1711
SET ENTRY VALUE AS NEXT FRAGMENT ADDRESS
FragmentAddress=Entry
S1706
"READING ON FRAGMENT BASIS" END

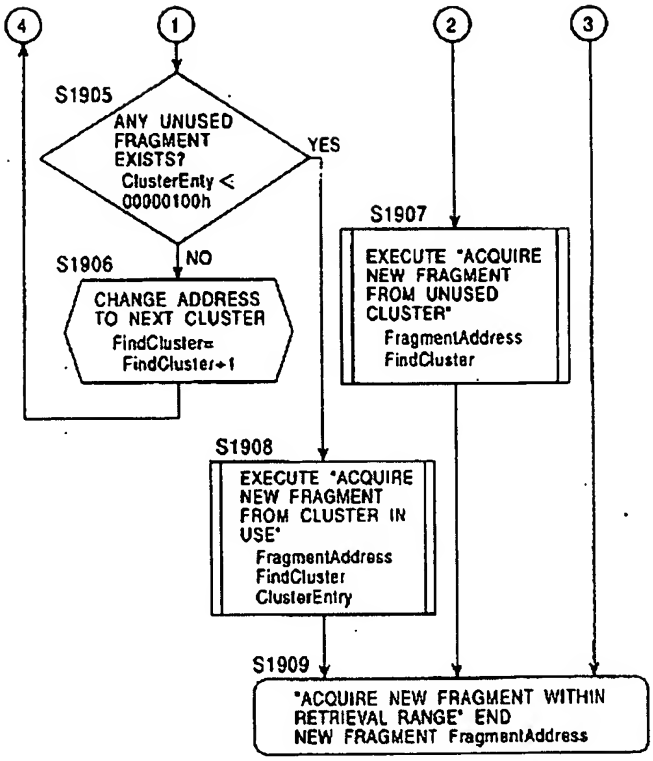
도면21


S1801
START "ACQUIRE NEW FRAGMENT"
CURRENT FRAGMENT FragmentAddress
ALLOCATION TYPE AllocType
S1802
HIERARCHICAL FILE?
AllocType!=0
YES
NO
S1803
EXECUTE "ACQUIRE NEW FRAGMENT WITHIN RETRIEVAL RANGE" FROM CURRENT CLUSTER TO FINAL CLUSTER
FragmentAddress
FindCluster=FragmemtAddress,High25Bit
FindEnd=TotalCluster - 1
S1804
EXECUTE "ACQUIRE NEW FRAGMENT OF HIERARCHICAL FILE"
FragmentAddress
AllocType
S1805
ACQUIRED?
YES
NO
S1806
EXECUTE "ACQUIRE FRAGMENT WITHIN RETRIEVAL RANGE" FROM FIRST TO CURRENT CLUSTERS
FragmentAddress
FindCluster=2
FindEnd=FragmentAddress, High25Bit - 1
S1807
ACQUIRED?
YES
NO
S1808
ERROR
S1809
"ACQUIRE NEW FRAGMENT" END
NEW FRAGMENT FragmentAddress

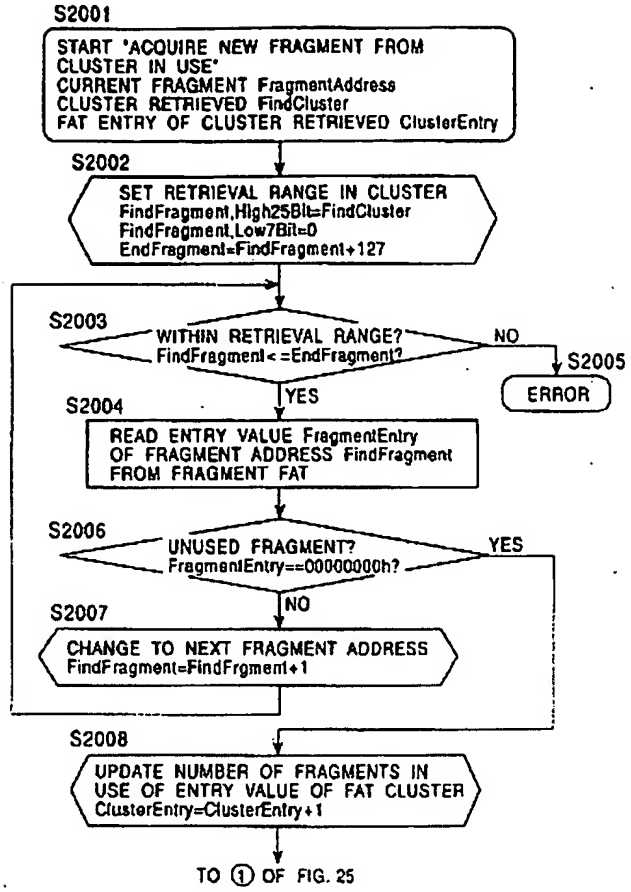
도면22


S1901
EXECUTE "ACQUIRE NEW FRAGMENT WITHIN RETRIEVAL RANGE"
CURRENT FRAGMENT FragmentAddress
START CLUSTER OF RETRIEVAL RANGE FindCluster
END CLUSTER OF RETRIEVAL RANGE FindEnd
S1902
WHTHIN RETRIEVAL RANGE? FindCluster < = FindEnd?
NO
YES
S1903
READ ENTRY VALUE ClusterEntry OF CLUSTER ADDRESS FindCluster FROM CLUSTER FAT
S1904
UNUSED CLUSTER? ClusterEntry== 00000000h
YES
NO
TO ① OF FIG. 23
TO ③ OF FIG. 23
FROM ④ OF FIG. 23
TO ② OF FIG. 23

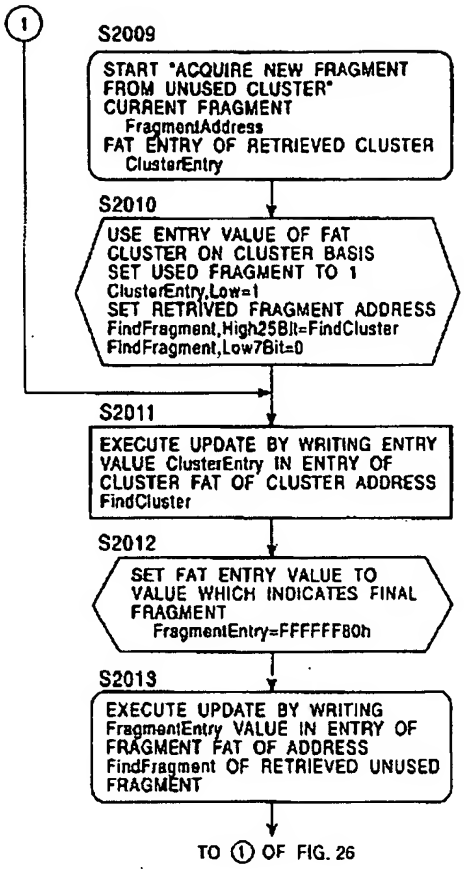
도면25


4
1
2
3
S1905
ANY UNUSED FRAGMENT EXISTS?
ClusterEnty < 00000100h
YES
NO
S1906
CHANGE ADDRESS TO NEXT CLUSTER
FindCluster=
FindCluster+1
S1907
EXECUTE "ACQUIRE NEW FRAGMENT FROM UNUSED CLUSTER"
FragmentAddress
FindCluster
S1908
EXECUTE "ACQUIRE NEW FRAGMENT FROM CLUSTER IN USE"
FragmentAddress
FindCluster
ClusterEntry
S1909
"ACQUIRE NEW FRAGMENT WITHIN RETRIEVAL RANGE" END
NEW FRAGMENT FragmentAddress

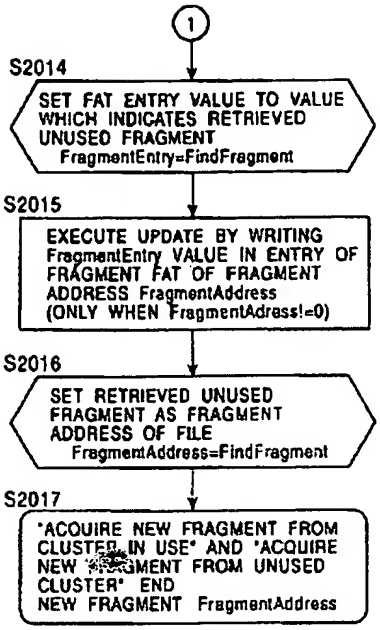
도면24


S2001
START 'ACQUIRE NEW FRAGMENT FROM CLUSTER IN USE'
CURRENT FRAGMENT FragmentAddress
CLUSTER RETRIEVED FindCluster
FAT ENTRY OF CLUSTER RETRIEVED ClusterEntry
S2002
SET RETRIEVAL RANGE IN CLUSTER
FindFragment,High25Bit=FindCluster
FindFragment,Low7Bit=0
EndFragment=FindFragment+127
S2003
WITHIN RETRIEVAL RANGE?
FindFragment<=EndFragment?
NO
S2005
ERROR
YES
S2004
READ ENTRY VALUE FragmentEntry OF FRAGMENT ADDRESS FindFragment FROM FRAGMENT FAT
S2006
UNUSED FRAGMENT?
FragmentEntry==00000000h?
YES
NO
S2007
CHANGE TO NEXT FRAGMENT ADDRESS
FindFragment=FindFrgment+1
S2008
UPDATE NUMBER OF FRAGMENTS IN USE OF ENTRY VALUE OF FAT CLUSTER
ClusterEntry=ClusterEntry+1
TO ① OF FIG. 25

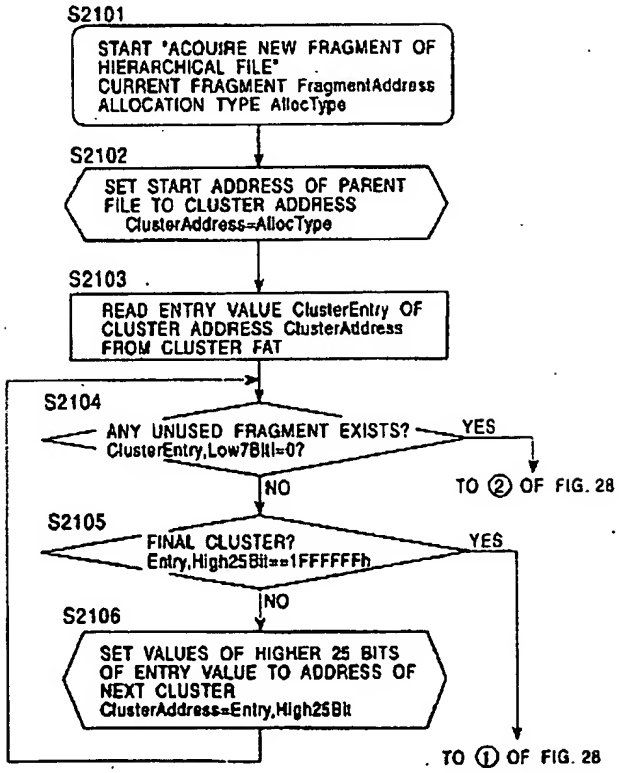
도면25


1
S2009
START "ACQUIRE NEW FRAGMENT FROM UNUSED CLUSTER" CURRENT FRAGMENT FragmentAddress FAT ENTRY OF RETRIEVED CLUSTER ClusterEntry
S2010
USE ENTRY VALUE OF FAT CLUSTER ON CLUSTER BASIS SET USED FRAGMENT TO 1 ClusterEntry,Low=1 SET RETRIVED FRAGMENT ADDRESS FindFragment,High25Bit=FindCluster FindFragment,Low7Bit=0
S2011
EXECUTE UPDATE BY WRITING ENTRY VALUE ClusterEntry IN ENTRY OF CLUSTER FAT OF CLUSTER ADDRESS FindCluster
S2012
SET FAT ENTRY VALUE TO VALUE WHICH INDICATES FINAL FRAGMENT FragmentEntry=FFFFFF80h
S2013
EXECUTE UPDATE BY WRITING FragmentEntry VALUE IN ENTRY OF FRAGMENT FAT OF ADDRESS FindFragment OF RETRIEVED UNUSED FRAGMENT
TO ① OF FIG. 26

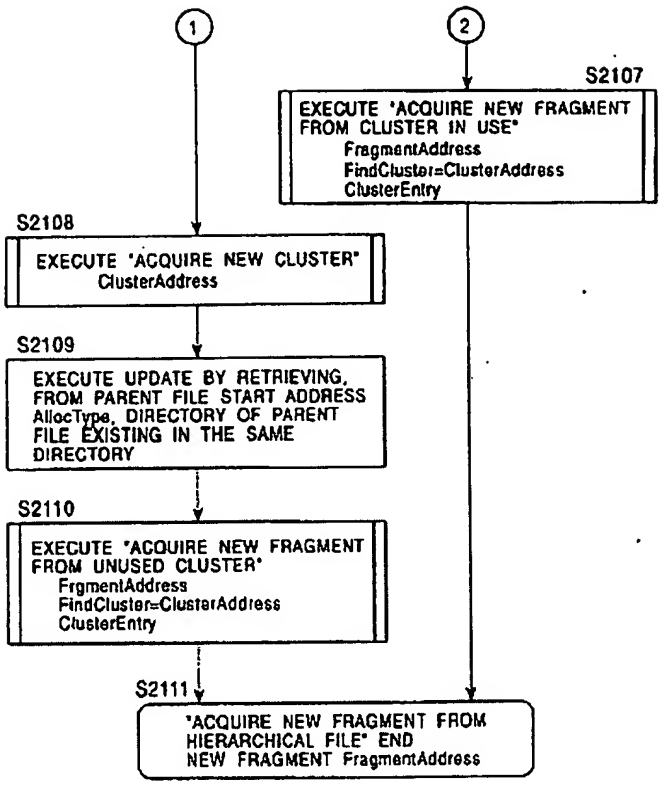
도면28


1
S2014
SET FAT ENTRY VALUE TO VALUE WHICH INDICATES RETRIEVED UNUSED FRAGMENT
FragmentEntry=FindFragment
S2015
EXECUTE UPDATE BY WRITING FragmentEntry VALUE IN ENTRY OF FRAGMENT FAT OF FRAGMENT ADDRESS FragmentAddress (ONLY WHEN FragmentAdress!=0)
S2016
SET RETRIEVED UNUSED FRAGMENT AS FRAGMENT ADDRESS OF FILE
FragmentAddress=FindFragment
S2017
"ACQUIRE NEW FRAGMENT FROM CLUSTER IN USE" AND "ACQUIRE NEW FRAGMENT FROM UNUSED CLUSTER" END
NEW FRAGMENT FragmentAddress

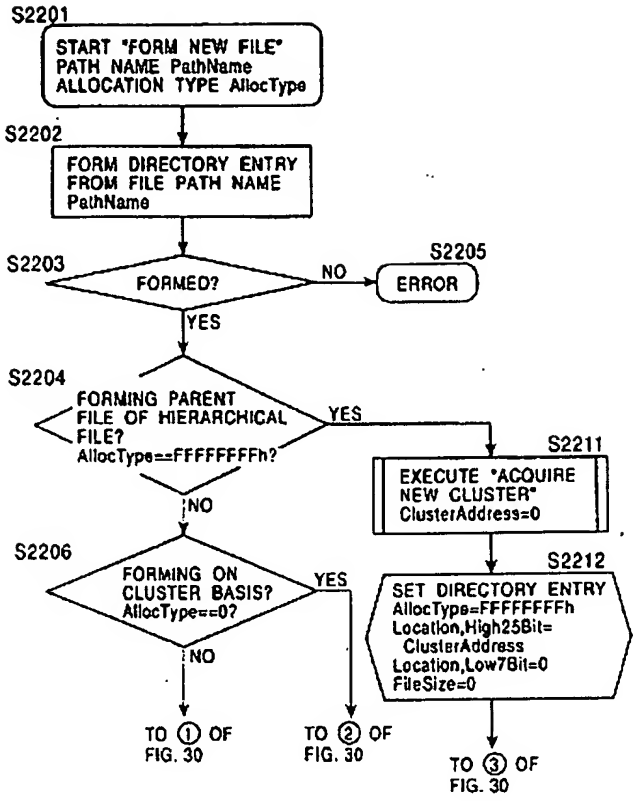
도면29


S2101
START "ACQUIRE NEW FRAGMENT OF HIERARCHICAL FILE"
CURRENT FRAGMENT FragmentAddress
ALLOCATION TYPE AllocType
S2102
SET START ADDRESS OF PARENT FILE TO CLUSTER ADDRESS
ClusterAddress=AllocType
S2103
READ ENTRY VALUE ClusterEntry OF CLUSTER ADDRESS ClusterAddress FROM CLUSTER FAT
S2104
ANY UNUSED FRAGMENT EXISTS?
ClusterEntry,Low7Bit=0?
YES
TO ② OF FIG. 28
NO
S2105
FINAL CLUSTER?
Entry,High25Bit==1FFFFFFh
YES
NO
S2106
SET VALUES OF HIGHER 25 BITS OF ENTRY VALUE TO ADDRESS OF NEXT CLUSTER
ClusterAddress=Entry,High25Bit
TO ① OF FIG. 28

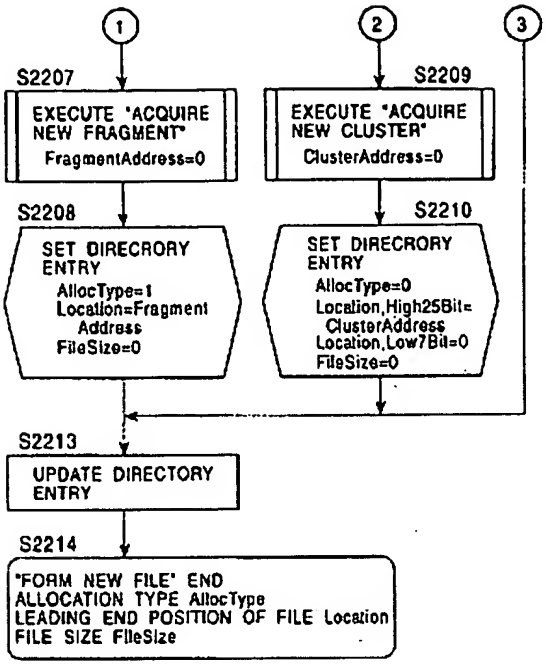
도면28


1
2
S2107
EXECUTE "ACQUIRE NEW FRAGMENT FROM CLUSTER IN USE"
FragmentAddress
FindCluster=ClusterAddress
ClusterEntry
S2108
EXECUTE "ACQUIRE NEW CLUSTER"
ClusterAddress
S2109
EXECUTE UPDATE BY RETRIEVING, FROM PARENT FILE START ADDRESS AllocType, DIRECTORY OF PARENT FILE EXISTING IN THE SAME DIRECTORY
S2110
EXECUTE "ACQUIRE NEW FRAGMENT FROM UNUSED CLUSTER"
FrgmentAddress
FindCluster=ClusterAddress
ClusterEntry
S2111
"ACQUIRE NEW FRAGMENT FROM HIERARCHICAL FILE" END
NEW FRAGMENT FragmentAddress

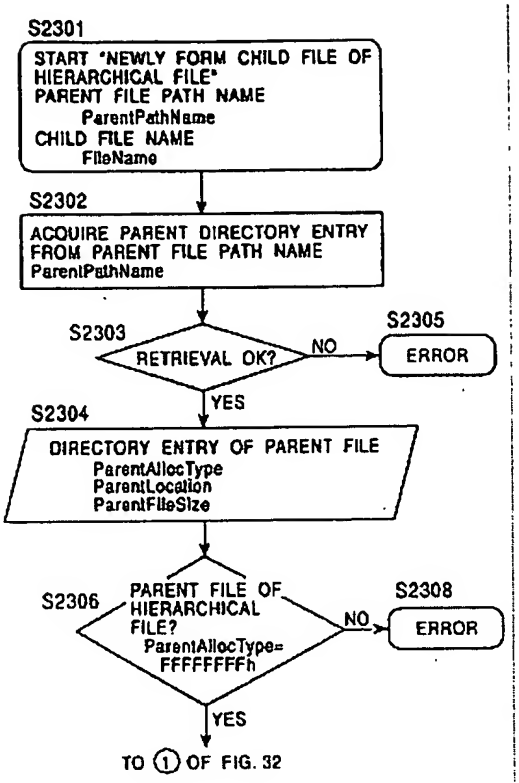
도면29


S2201
START "FORM NEW FILE"
PATH NAME PathName
ALLOCATION TYPE AllocType
S2202
FORM DIRECTORY ENTRY
FROM FILE PATH NAME
PathName
S2203
FORMED?
NO
S2205
ERROR
YES
S2204
FORMING PARENT
FILE OF HIERARCHICAL
FILE?
AllocType==FFFFFFFFh?
YES
S2211
EXECUTE "ACQUIRE
NEW CLUSTER"
ClusterAddress=0
NO
S2206
FORMING ON
CLUSTER BASIS?
AllocType==0?
YES
S2212
SET DIRECTORY ENTRY
AllocType=FFFFFFFFh
Location,High25Bit=
ClusterAddress
Location,Low7Bit=0
FileSize=0
NO
TO ① OF
FIG. 30
TO ② OF
FIG. 30
TO ③ OF
FIG. 30

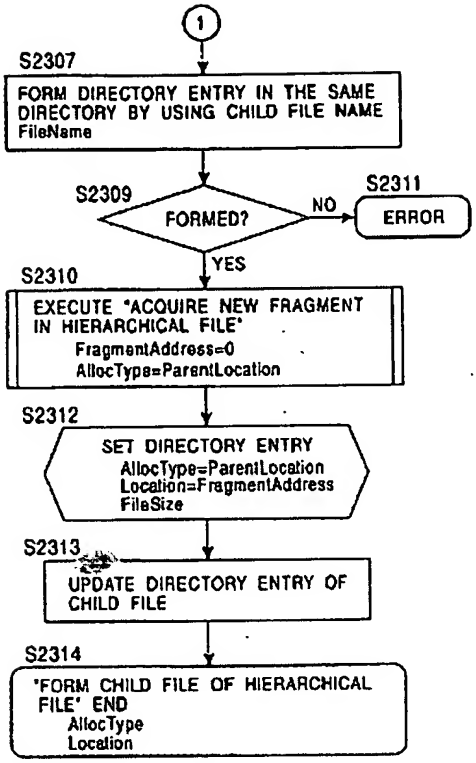
도면30


1
2
3
S2207
EXECUTE "ACQUIRE NEW FRAGMENT"
FragmentAddress=0
S2209
EXECUTE "ACQUIRE NEW CLUSTER"
ClusterAddress=0
S2208
SET DIRECRORY ENTRY
AllocType=1
Location=Fragment Address
FileSize=0
S2210
SET DIRECRORY ENTRY
AllocType=0
Location,High25Bit=ClusterAddress
Location,Low7Bit=0
FileSize=0
S2213
UPDATE DIRECTORY ENTRY
S2214
"FORM NEW FILE" END
ALLOCATION TYPE AllocType
LEADING END POSITION OF FILE Location
FILE SIZE FileSize

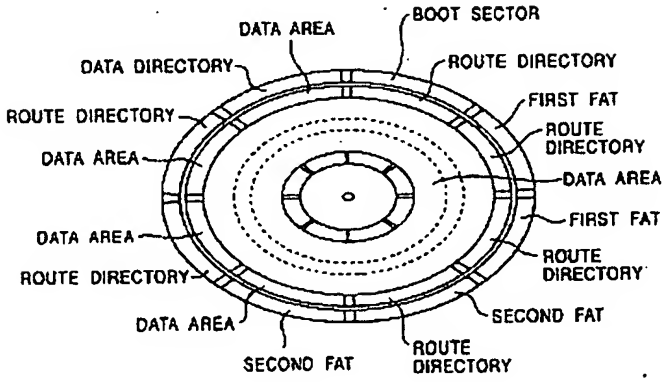
도면31


S2301
START "NEWLY FORM CHILD FILE OF HIERARCHICAL FILE"
PARENT FILE PATH NAME
ParentPathName
CHILD FILE NAME
FileName
S2302
ACQUIRE PARENT DIRECTORY ENTRY FROM PARENT FILE PATH NAME
ParentPathName
S2303
RETRIEVAL OK?
NO
S2305
ERROR
YES
S2304
DIRECTORY ENTRY OF PARENT FILE
ParentAllocType
ParentLocation
ParentFileSize
S2306
PARENT FILE OF HIERARCHICAL FILE?
ParentAllocType= FFFFFFFFh
NO
S2308
ERROR
YES
TO (1) OF FIG. 32

도면32


1
S2307
FORM DIRECTORY ENTRY IN THE SAME DIRECTORY BY USING CHILD FILE NAME FileName
S2309
FORMED?
NO
S2311
ERROR
YES
S2310
EXECUTE "ACQUIRE NEW FRAGMENT IN HIERARCHICAL FILE"
FragmentAddress=0
AllocType=ParentLocation
S2312
SET DIRECTORY ENTRY
AllocType=ParentLocation
Location=FragmentAddress
FileSize
S2313
UPDATE DIRECTORY ENTRY OF CHILD FILE
S2314
"FORM CHILD FILE OF HIERARCHICAL FILE" END
AllocType
Location

도면33


BOOT SECTOR
DATA AREA
DATA DIRECTORY
ROUTE DIRECTORY
FIRST FAT
ROUTE DIRECTORY
DATA AREA
DATA AREA
FIRST FAT
ROUTE DIRECTORY
ROUTE DIRECTORY
DATA AREA
SECOND FAT
ROUTE DIRECTORY
SECOND FAT
ROUTE DIRECTORY

도면34

101 SYSTEM REGION	
102	CLUSTER 2
103	3
104	4
105	5
106	6
107	7
108	8
109	9
110	10
111	11
112	12
113	N-1
114	N

도면35

201 FILE1.DAT		202 2	203 1254
204 FILE2.DAT		205 4	206 2013
207 FILE3.DAT		208 8	209 47

도면36

301	302	303(2) 3	304(3) 6
305(4) 7	306(5) 0	307(6)65535	308(7) 11
309(8)65335	310(9)65535	311(10) 0	312(11) 9
313(12) 0			
		314 0	315 9

도면37

401 4	402 T*2	403 7	404 T*3	405 11	406 T*1	407 9